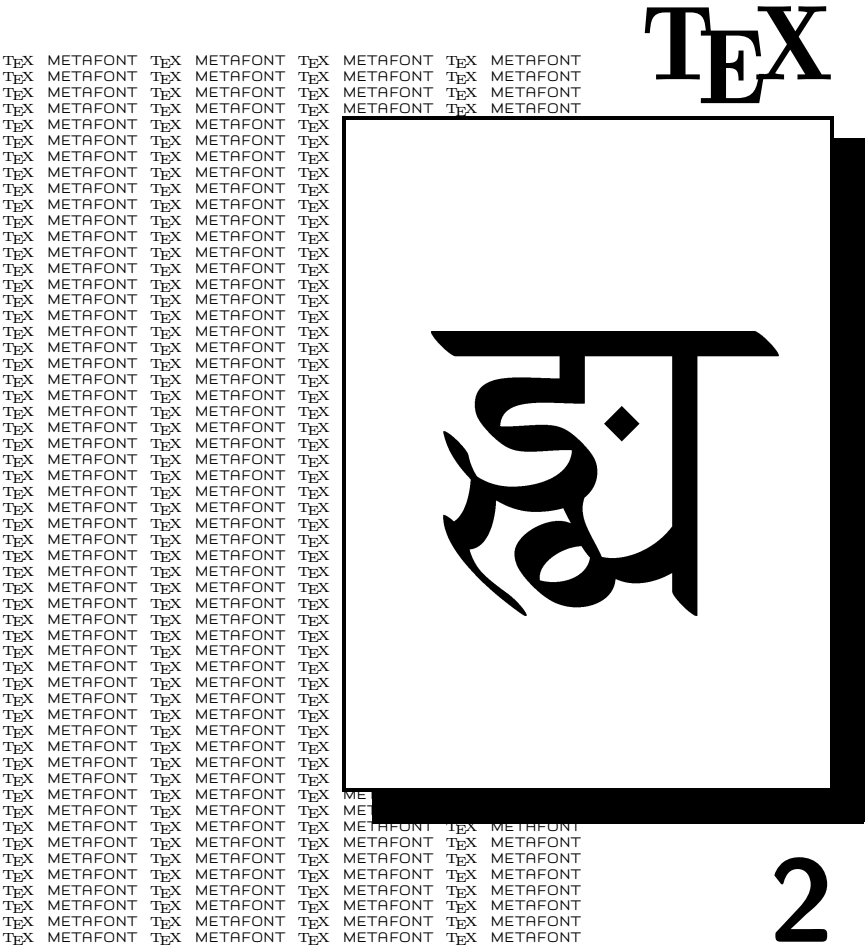


ZPRAVODAJ 96 2

ČESKOSLOVENSKÉHO SDRUŽENÍ UŽIVATELŮ TEXU

OBSAH

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 přístupných archívech.

Starší čísla Zpravodaje budou zveřejněna v elektronické podobě v nejbližší době. Vyzýváme autory, kteří s tímto způsobem zveřejnění svých článků nesouhlasí, aby o tom uvědomili redakci Zpravodaje.

Slovo úvodem

Petr Sojka

TUG and *You Together* We Can Do It
Michel Goossens, TUG President, 1996

Drazí TEXisté,

je mou milou povinností na tomto místě poděkovat dosavadnímu předsedovi CSTUGu Karlu Horákovi za to, že po více než tři roky třímal taktovku sdružení. Kromě předsednictví redigoval s pečlivostí a důkladností, byť s notným skluzem, občasník sdružení, zvaný Zpravodaj, jejž držíte v rukou.

Přes tyto nesnáze členská základna sdružení přesáhla 400 individuálních a 40 kolektivních členů a tento pozitivní trend se zatím neobrátil, což lze nesporně považovat za úspěch. V této době se TEX stal žádaným u celé plejády nakladatelů vyžadujících kvalitní sazbu vydávané knižní produkce[1]. V mnohých aspektech je tedy v další etapě činnosti sdružení na co navazovat.

Blahopřeji všem nově zvoleným členům výboru a přeji jim, aby jim počáteční entuziasmus dlouho vydržel a pokud neexistuje, aby vznikl. Zejména patří povzbuzení těm, kteří na sebe vzali *nezištně* příslib trvalejší služby. Mám na mysli zejména nového šéfredaktora a editora Zpravodaje Zdeňka Wagnera, jemuž vděčíte vysokou měrou za pravidelnost, s jakou letošní Zpravodaje vycházejí. Dalším příslibem je jihočeská nabídka na realizaci agendy sdružení s možným přístupem přes Internet od Ládi Lhotky. Libor Škarvada se podujal úkolu udržovat dokument „Často kladené otázky“, jenž by měl být nepostradatelným kompendiem CSTEXisty. Tuto neocenitelnou pomůcku obdržíte jakožto speciální číslo Zpravodaje 3/96 na podzim tohoto roku. Martin Bílý administruje distribuční elektronický list sdružení spolu s archivem cstugwaru přístupným členům přes kód PIN.

Nástin zamýšlených aktivit jednotlivých členů výboru jste ostatně měli možnost zhlédnout na volebních lístcích. Jaký ale CSTUG bude, nezáleží zcela jen na výboru, ale na Vás všech, jak se do činnosti naší organi-

[1] TEXem byl například sázen první díl Všeobecné encyklopedie, kterýžto projekt je deklarován autory jako „největší literární projekt konce století“.

zace zapojíte sami – například nabídnutím drobné výpomoci s překladem článku, vytvořením přehledového článku do Zpravodaje, zodpovídáním dotazů ostatních CSTEXistů na elektronických listech sdružení nebo řešením TEXnických problémů vašich spolupracovníků na vašich pracovištích. Počet a rozsah CSTUGem pořádaných přednášek, seminářů, srazů nebo školení bude záviset především na tom, zda se najdou aktivisté na těchto akcích schopní a ochotní participovat, ať již aktivně, nebo pasivně. Jen podobnými aktivitami a přístupem lze předejít tomu, aby se CSTUG nedostal do podobné krize, jakou řeší TUG, jehož členská základna se během posledních několika let snížila ze 4 000 členů na 1 200. Nárůstem členské základny sdružení lze předejít případným finančním problémům, jež by při současném soustavném trendu deficitního rozpočtu CSTUGu mohly přijít velmi brzy.

Sdružení přispělo na diskové kapacity ftp `ftp://ftp.cstug.cz/` a www `http://www.cstug.cz/cstug` serverů sdružení, jež jsou z větší části domovem na Fakultě Informatiky Masarykovy University v Brně. Teprve nedávno, jako jeden z prvních počinů nového výboru, se podařilo zřídit vlastní internetovskou doménu `cstug.cz`, byť bez vlastní výpočetní techniky. TEXovskou internetovskou komunitou je hojně využíváno zejména zrcadlo CTAN (Comprehensive TEX Archive Network), jehož kapacita dnes přesahuje 2 GB TEXového softwaru. V těchto intencích (`www` stránka sdružení atp.) se tento servis patrně bude rozšiřovat, pokud se ovšem členská základna neztotožní s názorem Petra Olšáka, že CSTUG se těmito aktivitami profiluje jako spolek „kybernetických veverek“. Jelikož posun od vytváření tištěných dokumentů k elektronickým je značný, a protože procento členů sdružení, jež jsou připojeni na Internet, stále roste, služby poskytované sdružením přes Internet hodlá nový výbor přinejmenším zachovat a tematice vytváření elektronických dokumentů se na těchto stránkách věnovat. Přehled nových elektronických adres a služeb v doméně `cstug.cz` najdete v tiráži.

Závěrem pak jedno přání osobní. Don Knuth, jehož jsme měli možnost poznat osobně nedávno na přednáškách v Brně a Praze, nazývá svoje dílo TEX přívlastkem „labour of love“[2]. Přeji Vám, aby v CSTUGu našel mnoho následovníků.

Petr Sojka
president@cstug.cz

[2] `ftp://ftp.cstug.cz/pub/tex/CTAN/systems/knuth/tex/tex.web.gz`

Zápis z valného shromáždění CSTUGu konaného 25. 5. 1996 v Praze, MFF UK, Sokolovská 83

1. Shromáždění zahájil Karel Horák. Informoval o nedávné „krizi“ ve sdružení a o práci výboru. Loňský skluz ve vydávání bulletinu bude napraven vydáním souhrnného čísla („superčísla“) za rok 1995. Jako editor tohoto čísla Karel Horák přislíbil jeho vydání ještě v měsíci červnu 1996.
2. Ludmila Olšáková přečetla zprávu o hospodaření CSTUGu v roce 1995. Ztráta cca 11 000 korun (při tisku jen 2 čísel bulletinu za rok) byla pokryta z rezerv sdružení. Do ztráty nebyla započítána cca stotisícová dotace na vydání publikace „Typografický systém TEX“ ani na nákup Rybičkovy knížky o LATEXu, neboť se předpokládá, že se investované prostředky vrátí prodejem.
3. Karel Horák přednesl zprávu o činnosti. Nejdůležitější akce CSTUGu za uplynulé období: přednáška Franka Mittelbacha, přednáška Donalda Knutha (v rámci jeho pozvání Fakultou informatiky), Zdeněk Wagner získán jako nový redaktor bulletinu.
4. Dále Karel Horák hovořil o nutnosti inventury hospodaření a o potřebě nového revizora.
5. Potom Petr Olšák mluvil o nové verzi balíku CSTEX a obšírně popisoval opravené a přetrvávající chyby. Dále informoval o svých aktivitách (diakritika v PostScriptu, čárové kódy) a zejména o svých plánech na další knížky o TEXu a o svých dalších potenciálních aktivitách.
6. K nejdůležitějším bodům programu valného shromáždění patřila volba nového výboru. Volební komise ve složení Dont, Obdržálek, Ulrych sečetla odevzdané hlasy a konstatovala, že všech 15 kandidátů bylo zvoleno.

jméno	hlasů	jméno	hlasů
M. Bílý	108	J. Rybička	95
K. Horák	77	P. Sekanina	96
J. Chlebíková	102	P. Sojka	114

pokračování

jméno	hlasů	jméno	hlasů
J. Krob	96	L. Škarvada	96
M. Kubeček	98	J. Veselý	115
L. Lhotka	107	Z. Wagner	111
K. Nemoga	89	J. Zlatuška	100
Š. Porubský	78		

Celkem bylo odevzdáno 120 vyplněných hlasovacích lístků a 3 lístky nevyplněné. Na jednom lístku byl připsán jako další kandidát Petr Olšák, který však obdržel pouze tento jediný hlas a nebyl tedy zvolen.

7. Dalším bodem programu byly přednášky Petra Sojky a Han The Thanha o PDF. Petr Sojka nejprve uvedl přehled existujících prohlížecích a manipulačních programů pro PDF. Čerstvý absolvent brněnské Fakulty informatiky MU Han The Thanh pak referoval o své diplomové práci, jejíž praktickou částí byl program `tex2pdf` – rozšíření TEXu o možnost zápisu vysázeného dokumentu ve formátu PDF.
   Následovala živá diskuse týkající se fontů, expanze virtuálních fontů, grafiky, komprese a křížových odkazů.
8. Petr Sojka shrnul diskusi, která proběhla v květnu v elektronické diskusní skupině výboru CSTUGu `cstugb-l`. Diskuse se týkala především situace CSTUGu a jeho fungování. Vyjádřil odhodlání překonat nedávné i současné potíže a načrtl plán činnosti CSTUGu na letošní rok i zamýšlený obsah letošních čísel bulletinu.
9. Jiří Veselý zopakoval připomínky ke Stanovám: zrušit bod 5 odstavce „Činnost...“, přeformulovat závěr bodu 2 odstavce „Členství...“. Tyto připomínky byly uznány za rozumné a valná hromada dala mandát komisi ve složení K. Horák, P. Sojka, J. Veselý, aby zpracovala finální verzi Stanov CSTUGu pro schválení na ministerstvu.
10. Valné shromáždění odhlasovalo výši členských příspěvků na letošní rok: letos zůstanou příspěvky na původní výši (150 korun individuální člen, 60 korun individuální člen – student, 1 000 korun kolektivní člen).
11. Různé:
    - Karel Horák tlumočil Jiřímu Rybičkovi zájem o jeho „LATEX pro začátečníky“.
    - Petr Šaloun: na Internetu bude (časem) vystavena diplomová

práce z VŠB řešící převod z RTF do TEXu.
- Jiří Veselý zmínil návrh Karola Nemogy koupit zařízení pro vypalování CD-ROM, aby bylo možno tímto způsobem šířit CSTEX.
- Jiří Rybička navrhl podpořit projekt vytvoření malé srovnávací studie (TEX vs. WYSIWYG programy pro DTP) určené laikům.
- Karel Horák informoval o svých zkušenostech se separací barev v METAPOSTu.
- Ladislav Lhotka oznámil, že v blízké době hodlá umístit databázi členů na WWW.
- Někdo z pléna navrhl projednat, jak lépe uložit peníze sdružení na účet s vyšší úrokovou sazbou, aby se snížily nemalé ztráty vzniklé inflací.
- Ladislav Lhotka se ptá, zda existuje kvalitní seznam českých/slovenských slov pro Ispell.

V Praze, dne 28. 5. 1996

Zapsal: Libor Škarvada

Ověřil: Petr Sojka

Po valné hromadě krátce zasedal nově zvolený výbor sdružení, aby jednomyslně zvolil nového předsedu – Petra Sojku, a dohodl datum příští výborové schůze (18. 6. 1996 v 10.30 v Brně).

Alternativní výstup programu TEX – PDF

Han The Thanh

Úvod

Systém TEX [4] je volně šiřitelný s úplnou dokumentací a díky tomu od doby vzniku TEXu už byla napsána řada programů, které jeho možnosti stále rozšiřují. Obsahem tohoto textu je popis rozšíření TEXu o novou možnost – vytváření dokumentů ve formátu PDF [2] přímo z TEXu. PDF

je formát souboru navržený firmou Adobe, se kterým pracují například produkty rodiny Adobe Acrobat, umí jej ale číst i další programy, například Ghostview. Účelem tohoto formátu a souvisejících programů je umožňovat uživateli snadno modifikovat a prohlížet elektronické dokumenty nezávisle na operačních systémech a aplikacích, ve kterých byly dokumenty vytvořeny.

V současné době existuje možnost vytvořit PDF dokumenty z TEXu přes PostScript [3] pomocí programů `dvips` (konvertuje z `DVI` do PostScriptu) a Acrobat Distiller (konvertuje z PostScriptu do PDF). Existuje ještě několik balíků maker v TEXu, které umožňují z TEXu vložit do `DVI` výstupu speciální sekvence. Tyto sekvence se vloží do PostScriptu pomocí programu `dvips` a výsledný PostScriptový program se ještě transformuje dalším programem, který konvertuje tyto speciální sekvence do formy PostScriptového operátoru `/pdfmark`. Nakonec program Acrobat Distiller na základě výskytů operátoru `/pdfmark` v PostScriptovém programu vytvoří požadované prvky z TEXu, např. hypertextové odkazy, záložky (bookmarks) nebo textové poznámky. Tento proces je celkem zdlouhavý a složitý, navíc se nelze obejít bez programu Acrobat Distiller, který je komerční a je dodáván pouze pro některé platformy. Další nevýhodou tohoto postupu je, že nedává optimální PDF výstupy, případně zabrání vytvořit složitější prvky dokumentu jako víceřádkové hypertextové odkazy. Z těchto důvodů bylo navrženo rozšíření programu TEX tak, aby produkoval PDF soubor místo klasického formátu `DVI`. To vychází z možnosti, která byla předložena autorem TEXu, profesorem Donaldem Ervinem Knuthem, ale zatím zůstala nerealizována. Program byl napsán pod vedením prof. Jiřího Zlatušky na Fakultě informatiky Masarykovy univerzity v Brně v rámci diplomové práce [5].

Přenositelný formát dokumentu – Úvod

V této části stručně vysvětlíme základní pojmy, vlastnosti a použití PDF.

Co je PDF?

PDF (Portable Document Format) je formát souboru, který se používá k prezentaci elektronického dokumentu nezávisle na aplikačním software, hardware a operačních systémech.

PDF dokument obsahuje jednu či více stránek. Každá stránka v dokumentu může obsahovat libovolnou kombinaci textu, grafiky a obrazů ve formátu nezávislém na výstupním zařízení a jeho rozlišení. PDF dokument také může obsahovat informace, které jsou užitečné (a použitelné) pouze v elektronické prezentaci, např. hypertextové odkazy. Kromě prezentace dokumentu PDF soubor navíc obsahuje další informace, např. verzi PDF, informaci o umístění důležitých struktur v souboru pro efektivní vyhledávání apod.

Hlavní rysy PDF

Uvedeme zde některé důležité vlastnosti PDF.

PostScriptový kreslicí model (PostScript imaging model)

PDF reprezentuje text a grafiku použitím podobného modelu jako v PostScriptu. Stejně jako PostScriptový program PDF sestavuje popis stránky (page description) tím, že vybarvuje vybrané oblasti.

PDF značkovací operátory (marking operator) jsou velmi podobné PostScriptovým. Hlavní rozdíl od PostScriptu je ten, že PDF není programovací jazyk a neobsahuje procedury, proměnné a řídící struktury. PDF v podstatě vyměňuje redukovanou flexibilitu za zlepšenou efektivitu. Typický PostScriptový program definuje množinu high-level operátorů použitím základních značkovacích operátorů. Na rozdíl od toho PDF definuje svou vlastní množinu high-level značkovacích operátorů, které jsou dostačující pro popisy většiny stránek. Tyto operátory jsou implementovány přímo ve strojovém kódu aplikací a ne v PostScriptovém kódu, a proto mohou být PDF popisy stránek zobrazovány mnohem rychleji. Navíc, díky omezení programovacích struktur v PDF, prohlížeče mohou mnohem efektivněji a spolehlivěji přistupovat k textům v PDF dokumentu.

Přenositelnost

PDF soubory mohou použít sedmibitové ASCII znaky pro reprezentaci dokumentu včetně bitových vzorků a speciálních znaků. Díky tomu jsou PDF dokumenty přenositelné i mezi nejrůznějšími platformami a operačními systémy. Programy Acrobat Reader a Aladdin Ghostview, které

PDF soubory umí zobrazovat, jsou volně šiřitelné pro všechny běžné platformy.

Komprese dat

PDF podporuje několik standardních kompresních formátů pro redukci velikosti PDF souboru:

- JPEG komprese barevných obrazů.
- CCITT Group 3, CCITT Group 4, LZW (Lempel-Ziv-Welch) a Run Length komprese černobílých obrazů.
- LZW komprese textu a grafiky.

Použitím JPEG komprese mohou být barevné obrazy komprimovány v poměru 10:1 a více. Účinnost komprese černobílých obrazů závisí na použitém formátu a vlastnostech obrazů, avšak redukce 2:1 až 8:1 je běžná. LZW komprese textu a grafiky dává redukci kolem 2:1. Všechny tyto formáty jsou tvořeny binárními daty, která se mohou konvertovat do ASCII 85-znakového kódování pro zachování přenositelnosti.

Nezávislost na fontu

Správa fontu je vždy podstatný a těžký problém v oblasti přenosu elektronických dokumentů. Obecně platí, že příjemce musí mít stejné fonty, které použil odesílatel při tvorbě dokumentu. Jinak je použit místo chybějícího fontu implicitní font, což může způsobit neočekávané a nežádoucí následky, protože implicitní font může mít jiné metriky znaků než původní. Odesílatel by mohl vložit použité fonty do PDF souboru, ale to může způsobit, že velikost poměrně krátkého dokumentu, např. dvoustránkový text se čtyřmi fonty, bude zabírat 10–250 kB. Jiná možnost je konvertovat všechny stránky do bitových obrazů pevného rozlišení. I v tomto případě velikost každé stránky po kompresi může být dost velká (45–60 kB při rozlišení 200 dpi). Navíc tato metoda způsobí ztrátu významu textu v dokumentu, čímž znemožňuje příjemci vyhledat nebo extrahovat text z dokumentu.

PDF poskytuje nové řešení tohoto problému, které definuje tvar dokumentu částečně nezávisle na fontech v něm použitých. PDF soubor pro každý použitý font obsahuje *popis fontu* (font descriptor), který se skládá z nejdůležitějších informací o fontu: metriky znaků, informace o stylu fontu apod. Tyto informace jsou použity při simulaci chybějícího

fontu, a zabírají typicky 1–2 kB pro každý font. Není-li požadovaný font k dispozici, potom se na základě informací v popisu fontu generuje náhradní font pro simulaci originálního fontu tak, aby se zachoval celkový vzhled a formát dokumentu.

Jednoprůchodové generování

Někdy je pro implementaci programů generujících PDF soubory žádoucí vytvořit PDF dokumenty v jednom průchodu. Důvodem může být systémové omezení a efektivita, např. omezená paměť, nebo nelze vytvořit dočasné pracovní soubory. Pro tento účel PDF podporuje jednoprůchodové generování pomocí mechanismu *nepřímých objektů* (indirect objects). Tento mechanismus umožňuje např. specifikovat délku nějakého objektu až *za* umístěním tohoto objektu.

Náhodný přístup k dokumentu

Programy, které čtou a zobrazují jednu stránku dokumentu v PostScriptu, musí projít celý PostScriptový soubor od začátku, dokud nenajdou požadovanou stránku. Proto čas potřebný na zobrazení jedné stránky v PostScriptu závisí nejen na složitosti zobrazované stránky, ale také na počtu stránek v dokumentu. To je značný problém při interaktivním prohlížení, kdy je důležité, aby čas potřebný na zobrazení jedné stránky byl nezávislý na celkovém počtu stránek.

Každý PDF soubor obsahuje *tabulku odkazů* (cross-reference table), která se používá pro zjištění umístění stránek a pro přímý přístup k nim, případně k dalším důležitým objektům. Tabulka odkazů je umístěna na konci PDF souboru, což dovoluje těm aplikacím, které generují PDF soubory v jednom průchodu, snadno ji uložit, a těm, které PDF soubory čtou, snadno ji najít. S použitím tabulky odkazů může být čas potřebný na zobrazení jedné stránky téměř nezávislý na jejich celkovém počtu.

Modifikace přidáním

PDF umožňuje přidat změny v dokumentu na konec souboru a nechat původní obsah beze změn. Přitom připojená data obsahují pouze nové nebo změněné objekty a obnovenou tabulku odkazů. Takto závisí čas modifikace PDF souboru na rozsahu modifikace, a ne na celkové veli-

kosti souboru. Navíc tento mechanismus dovoluje odstranit změny velice snadno – smazáním přidaných dat.

Rozšiřitelnost

PDF je navržen tak, aby byl snadno rozšiřitelný. Vývojáři určitě budou chtít přidat do PDF vlastnosti, které dosud nejsou implementovány nebo vymyšleny, např. zvuková data. PDF je také zpětně kompatibilní. To znamená, že aplikace, které pracují s předchozími verzemi PDF, zachovávají (alespoň částečně) svou funkčnost na souborech novější verze s vlastnostmi, které tyto aplikace dosud neimplementují; tyto vlastnosti by měly být jednoduše ignorovány.

PDF a PostScript

I když PDF a PostScript jsou velice podobné, mají jisté odlišnosti. Tyto odlišnosti jsou dány účelem a použitím, pro které byly formáty navrženy.

- PDF soubor může obsahovat prvky, např. hypertextové odkazy, které jsou užitečné jen při interaktivním prohlížení.
- Z důvodu efektivity PDF nepodporuje žádné konstrukce programovacího jazyka.
- PDF má pevně definovanou strukturu souboru a dokumentu, což umožňuje rychlý přístup k libovolné části dokumentu, případně snadnou modifikaci dokumentu.
- PDF soubory obsahují informace o fontu, které zajišťují věrné zobrazení dokumentu.

Protože mezi PDF a PostScriptem jsou určité rozdíly, nelze PDF soubory tisknout na PostScriptové tiskárně přímo, ale musí se provést následující kroky:

- Vložení *procsets* – množiny procedur, které implementují PDF značkovací operátory pro popis stránky, do PostScriptového programu.
- Extrahování obsahu každé stránky, protože v PDF souboru stránky nemusí být umístěny sekvenčně.
- Dekódování komprimovaných dat textu, grafiky a obrazů. Tento krok není nutný pro tiskárny s jazykem PostScript Level 2, které mohou akceptovat komprimovaná data.

- Vložení použitých zdrojů (resources), např. fontů, do PostScriptového souboru. Náhradní fonty se generují na základě informace obsažené v popisu fontu.
- Umístění dat v korektním pořadí.

Výsledný soubor je tradiční PostScriptový program, který plně prezentuje vzhled dokumentu, ale neobsahuje PDF prvky jako hypertextové odkazy, textové poznámky a záložky.

Použití PDF

PDF soubory lze vytvořit buď z aplikací, nebo konvertováním z PostScriptu. Mnohé aplikace mohou vytvářet přímo PDF soubory pomocí programu PDF Writer, který je dodáván pro Apple Macintosh i na počítače s operačním systémem MS Windows. PDF Writer funguje jako ovladač tiskárny, čte příkazy z výstupu jiných ovladačů tiskáren (QuickDraw pro Macintosh a GDI pro Windows) a na výstupu generuje PDF soubory místo příkazů pro tiskárnu.

Některé programy vytvářejí přímo PostScriptové soubory, které lze konvertovat do PDF pomocí programu Acrobat Distiller. Acrobat Distiller dokáže generovat efektivnější PDF soubory než PDF Writer pro některé aplikace.

Pro prohlížení a tisk PDF souboru jsou přístupné dva programy – Acrobat Reader a Acrobat Exchange. Uživatel může prohlížet dokument použitím hypertextových odkazů, záložek a zmenšených obrazů stránek. Textové řetězce z dokumentu lze vyhledat, případně použít i v jiných aplikacích. Navíc Acrobat Exchange dovoluje změnit PDF dokumenty přidáním textových poznámek, hypertextových odkazů, záložek a zmenšených obrazů stránek.

V poslední době vzrůstá zájem o PDF, zvlášť o tzv. Amber generaci produktů rodiny Adobe Acrobat. Cílem je prosadit PDF a optimalizovat použití PDF na Internetu.

TEX a WEB

Program TEX je velmi dobře dokumentován díky tomu, že byl napsán v systému WEB, který vymyslel sám autor TEXu pro tvorbu dobře dokumentovaných programů. WEB umožňuje psát do jediného souboru jednak zdrojový text programu v programovacím jazyce Pascal (s některými

omezeními), jednak dokumentaci k programu v TEXu. Existují i další systémy WEB, které podporují jiný programovací jazyk než PASCAL a jiný formátovací jazyk než TEX, ale touto problematikou se zde zabývat nebudeme.

Systém WEB obsahuje dva programy. První program, tangle, vytvoří ze souboru napsaného ve WEBu (s příponou .web) zdrojový text programu v PASCALu. Přitom WEB podporuje některé další vlastnosti, které PASCAL neposkytuje, např. možnost definovat makro, lepší zpracování řetězců, nebo rozdělení programu do sekcí, které potom program tangle sestaví dohromady ve správném pořadí. Druhý program, weave, vytvoří zdrojový soubor v TEXu, který obsahuje dokumentaci a výpis programu.

Důležitá vlastnost systému WEB je ta, že programy napsané ve WEBu jsou snadno modifikovatelné. Modifikace přitom může probíhat bez zásahu do původního souboru web pomocí mechanismu *změnového souboru* (change file). Změnový soubor obsahuje popis, které řádky je nutno zaměnit novými. Oba programy tangle a weave dokáží číst kromě souborů web i změnové soubory (s příponou .ch).

Další zajímavost na programu TEX je, že většina instalací TEXu v dnešní době stále používá původní zdrojový soubor tex.web, který napsal autor TEXu před mnoha lety. K instalaci TEXu na nejrůznějších systémech byl vytvořen balík web2c, který obsahuje všechny potřebné soubory pro instalaci, včetně podpůrných programů pro konverzi zdrojového textu v PASCALu do jazyka C, a další systémové změny. Díky tomu, že programovací jazyk podporovaný WEBem je velice omezený, lze provozovat TEX z původního zdrojového soubor tex.web téměř na všech systémech. Lokální modifikace je provedena pomocí změnových souborů a dalších programů v balíku web2c.

dvi2pdf versus tex2pdf

Myšlenku vytvořit PDF z TEXu můžeme implementovat buď jako DVI ovladač dvi2pdf, nebo jako upravenou verzi originálního programu TEX na tex2pdf. Každý postup má své výhody i nevýhody.

Výhody dvi2pdf

- Máme velkou volnost při implementaci. Můžeme tedy zvolit svůj oblíbený programovací jazyk, svůj postup při programování, svoje metody a styly psaní atd.

- Kompilace může být jednoduchá a rychlá, hledání chyb a ladění je také snadnější.
- `DVI` formát je poměrně jednoduchý a přehledný.
- Existuje velké množství `DVI` ovladačů, které mohou být podkladem pro začátek naší práce.
- Nemusíme zasahovat do původního programu TEX.

Nevýhody `dvi2pdf`

- Nutnost znovu implementovat některé složité algoritmy, např. zpracování `TFM` metrik, aritmetiku s typem *scaled* atd., které už jsou implementovány (a dobře testovány) v TEXu.
- Museli bychom vyřešit problém přenositelnosti. Přenositelnost je nepříjemná technická záležitost, která vyžaduje spoustu úsilí a spolupráci mezi programátory na různých systémech.
- Pokud jde o implementaci „neTEXovských“ prvků (hypertextových odkazů, záložek), museli bychom vložit z TEXu do `DVI` souboru speciální sekvence, které označují pozice hypertextových odkazů a další potřebné informace. Potom program `dvi2pdf` na základě těchto sekvencí bude generovat příslušné objekty do PDF. Tato cesta nám neumožňuje využít řadu důležitých informací o dokumentu, které TEX poskytuje, např. o strukturách boxů v dokumentu. Kdybychom např. chtěli mít víceřádkový hypertextový odkaz, který obsahuje několik vět v odstavci, a chceme, aby výskyt hypertextového odkazu neovlivnil sazbu odstavce, museli bychom nejdříve sázet tento odstavec bez vložení hypertextového odkazu. Potom prohlížíme sazbu a ručně přidáme na příslušná místa v textu sekvence, které nám označují pozice, kde chceme mít hypertextový odkaz. Protože ve chvíli, kdy píšeme texty v TEXu, neznáme řádkové zlomy, nemůžeme určit, kam máme tyto sekvence napsat, abychom dostali požadovaný výsledek bez prohlížení dokumentu po sazbě. Při použití tohoto postupu máme dvě možnosti. Buď budeme muset sázet hypertextové odkazy do boxu a tím zakazujeme řádkový zlom uvnitř hypertextového odkazu, nebo budeme muset hypertextové odkazy ručně přidávat na příslušná místa, pokud chceme mít řádkový zlom uvnitř hypertextového odkazu.

Výhody `tex2pdf`

- Možnost využít řadu algoritmů a rutin, které už byly napsány a dobře testovány v původním programu.
- Není to nejjednodušší řešení z implementačního hlediska, ale je přehledným, čistým a otevřeným řešením, které může být využíváno na různých platformách.
- Co se týká přenositelnosti, díky své dlouhodobé existenci program TEX už má přenositelnost vyřešenu pro většinu systémů pomocí mechanismu změnového souboru a balíku `web2c`. O přenositelnost se tedy nebudeme muset vůbec starat. Použitím nového změnového souboru naše implementace bude přenositelná stejně jako původní program TEX. Tím ušetříme spoustu času a úsilí. Navíc instalace `tex2pdf` nebude složitější než instalace TEXu z balíku `web2c`.
- Umožňuje vyřešit problém víceřádkových hypertextových odkazů, který nelze řešit pomocí `dvi2pdf`. Můžeme využít řadu informací z TEXu, např. struktury a velikosti boxů. Díky tomu, že všechny texty v TEXu jsou sázeny pomocí boxů, lze např. poměrně snadno zjistit, které boxy obsahují text, který chceme mít označený jako hypertextový odkaz. To v praxi znamená, že stačí označit, kde začíná a kde končí text, který odpovídá hypertextovému odkazu. Potom pro všechny boxy, které obsahují tento text, generujeme odpovídající hypertextové odkazy. Tento postup nijak neovlivňuje sazbu dokumentu a umožňuje velmi snadno označit složité prvky jako víceřádkové hypertextové odkazy apod. Další výhodou tohoto řešení je, že ze struktur boxů lze zjistit mnoho implicitních informací. Proto syntaxe nových primitivů mohou být poměrně jednoduché.
- `tex2pdf` lze poměrně snadno kombinovat s dalšími rozšířeními TEXu, která jsou implementována stejným mechanismem (tedy pomocí změnového souboru k původnímu programu TEX), např. ε-TEX.

Nevýhody `tex2pdf`

- Nutnost zasahovat a přidávat nové primitivy do TEXu. Pomocí speciálních sekvencí přes primitiv `\special` nelze vložit do PDF souboru objekty jako hypertextové odkazy, bookmarky a textové

poznámky, protože tyto objekty nemohou být součástí PDF popisu stránky, ale musí být specifikovány odděleně s popisem stránky.
- Závislost na původním programu je také velkým problémem. Museli bychom studovat zdrojový text programu a psát podle stylu a v jazyce, ve kterém byl program napsán. Jazyk podporovaný WEBem je také velice omezený.
- Instalace pomocí balíku `web2c` má poměrně zdlouhavou a složitou kompilaci, která má několik fází. V první fázi je nutné vytvořit zdrojový text programu v PASCALu `tex.p` pomocí souborů `tex.web` a `tex.ch`. V další fázi je třeba konvertovat soubor `tex.p` do jazyka C pomocí podpůrných programů z balíku `web2c`, a teprve potom můžeme kompilovat zdrojové texty v jazyce C do spustitelného formátu. Tento postup nelze nijak zkrátit a může být velkým problémem při vývojové fázi, zvláště když kompilace probíhá na počítačích, které nejsou dostatečně výkonné.
- Hledání chyb a ladění programu je velice obtížné a nepřehledné.

Celkově `tex2pdf` je implementačně náročnější úkol, ale současně je čistším řešením, které je přehledné, přenositelné a otevřené pro další využití, případně rozšíření. Proto byl program napsán jako `tex2pdf`, tedy jako změnový soubor k původnímu programu TEX.

Program `tex2pdf`

Zde popisujeme základní rysy programu `tex2pdf`.

Nový primitiv `\pdfoutput`

Primitiv `\pdfoutput` byl přidán jako nový celočíselný parametr, jehož kladná hodnota specifikuje výstup do PDF souboru, v opačném případě je výstupem klasický `DVI` soubor. Pokud chceme mít výstup do PDF formátu, musíme nastavit hodnotu tohoto parametru na kladnou hodnotu *předtím*, než TEX vypíše první stránku na výstup.

Správa fontů

PDF podporuje tři typy fontu: Type1, TrueType a Type3 (uživatelsky definovaný) font. `tex2pdf` zatím podporuje pouze Type1 fonty [1], protože v současné době je možno získat `TFM` metriky jen pro Type1 fonty

pomocí programu `afm2tfm`. Použití Type3 fontů zatím není podporováno, v budoucnu bude možno pomocí Type3 fontu vložit bitmapové fonty do PDF výstupu. Pro TrueType fonty nelze (zatím) získat `TFM` metriky (neexistuje žádný nekomerční produkt, který by generoval `TFM` metriky z TrueType fontů).

`tex2pdf` dokáže generovat redukované fonty pro Type1 fonty. Tyto fonty jsou totožné s původními fonty kromě toho, že obsahují jen definice znaků, které byly v dokumentu použity. Pokud Type1 fonty jsou vloženy celé do PDF souboru, a přitom nejsou použity všechny znaky ve fontu, jsou mnohé informace nadbytečné. Navíc je použití redukovaných fontů někdy nutné, protože si někteří výrobci fontů velice potrpí na to, aby jejich fonty nebyly vloženy do dokumentu celé, ale smí tam být pouze redukovaná verze fontu. Generování redukovaných fontů může poměrně efektivně zmenšit velikost výstupního PDF souboru pro krátké dokumenty, ale s mnoha použitými fonty.

Další důležitá část v oblasti správy fontů je správa virtuálních fontů. Je to mechanismus, který vymyslel autor TEXu pro rozšíření možností použití fontů v TEXu. Jedná se o popis ve `VF` (Virtual Font) formátu, který je velice podobný `DVI` formátu. Ve `VF` souboru je specifikována realizace každého znaku, nejčastěji je to kompozice několika znaků. `VF` font je tedy jakýsi neskutečný (virtuální) font, který jen popisuje způsob realizace každého znaku ve fontu. I když mechanismus virtuálních fontů neexistuje dlouho, jeho použití je už tak rozšířené, že téměř všechny `DVI` ovladače tyto fonty akceptují. Virtuální fonty jsou zvlášť významné pro sazbu v neanglických jazycích, např. v češtině, kdy se používají k přemapování kódování a kompozici akcentovaných znaků.

Správa virtuálních fontů v podstatě znamená implementovat interpretaci `DVI` příkazů do PDF značkovacích operátorů. Podpora virtuálních fontů je nezbytná, protože většina Type1 fontů neobsahuje všechny znaky množiny ISOLatin2, ale obvykle jen znaky anglické abecedy a další akcenty pro kompozici akcentovaných znaků. Abychom mohli použít takový font např. pro sazbu v češtině, musíme použít virtuální font, který má české znaky popsané jako kompozice znaků a akcentů z původního fontu.

Zatím jsou virtuální fonty implementovány pomocí expanze `DVI` příkazů, což znamená mnohem menší efektivitu pro texty, kde je mnoho kompozitních znaků. Do budoucna lze definovat virtuální fonty jako Type3 fonty v PDF, což teoreticky může zlepšit použití virtuálních fontů.

Type1 fonty a virtuální fonty musí být umístěny v adresářích, kde TEX hledá `TFM` metriky (tedy v adresářích specifikovaných v proměnných prostředí, například pod Unixem to jsou proměnné `TFMFONTS`, `TEXFONTS` a `TEXTFMS`).

Implementace hypertextových odkazů, textových poznámek a záložek

Jak bylo řečeno, implementace hypertextových odkazů, textových poznámek a záložek se neobejde bez přidání nových primitivů do TEXu. Tyto nové primitivy lze poměrně přehledně a snadno implementovat díky tomu, že autor TEXu napsal program tak, aby byl snadno rozšiřitelný. Hlavní činnost programu TEX je konstruovat systém boxů, který popisuje vzhled každé stránky. Tento systém boxů je reprezentován jako strom, jehož kořenem je vnější box. Uzly tohoto stromu ale nejsou jen boxy, ale také mezery mezi boxy, penalty a další data. TEX pro další rozšíření má zvlášť jeden typ uzlu, který se nazývá *whatsit* uzel. Pomocí whatsit uzlů lze TEX snadno bez komplikací rozšířit. Primitivy byly implementovány tak, aby měly co nejméně parametrů. Většina parametrů má implicitní hodnoty. Zde popisujeme nově přidané primitivy.

- `\pdfannottext` implementuje textovou poznámku. Má pouze jeden povinný parametr, který specifikuje vlastní text v poznámce. Textová poznámka se vyskytuje na místě odpovídajícím výskytu primitivu `\pdfannottext` ve zdrojovém textu v TEXu.
- `\pdfannotlink` a `\pdfendlink` implementuje hypertextový odkaz. Za `\pdfannotlink` lze specifikovat rozměry odkazu stejným způsobem jako u primitivů `\hrule` a `\vrule` v TEXu, tj. pomocí klíčových slov `depth`, `height` a `width`. Pokud některý z nich není uveden, bude jeho hodnota převzata z rozměrů boxu, který odkaz obsahuje. Poslední povinný parametr specifikuje *klíč* odkazu, což je číslo, které slouží pro navázání spojení mezi cílem a odkazem. Odpovídající cíl a odkaz budou mít stejný klíč. Pokud má odkaz klíč, který neodpovídá žádnému cíli, program skončí s chybou bez generování PDF souboru. Ke každému výskytu primitivu `\pdfannotlink` musí existovat odpovídající výskyt primitivu `\pdfendlink`, který označuje, kde končí odkaz. Mezi výskytem primitivu `\pdfannotlink` a odpovídajícím

výskytem primitivu `\pdfendlink` nesmí být žádný další výskyt primitivu `\pdfannotlink`, tj. odkazy nesmějí být do sebe vnořeny. Další podmínkou je to, že výskyty primitivu `\pdfannotlink` a odpovídající `\pdfendlink` musí být v boxech, které jsou obsaženy přímo v jednom boxu, a musí být stejného typu. Tedy oba boxy jsou buď `\vbox` nebo `\hbox`. Tato podmínka souvisí se zjištěním boxů, které obsahují odkaz. Pokud jsou výskyty obsaženy v hboxech, potom také všechny hboxy mezi nimi obsahují tento odkaz. Podobně to platí i pro vbox. Další nepovinné parametry, které určují horizontální rádius, vertikální rádius a šířku obruby odkazu, lze specifikovat za klíčovým slovem `border`.

- `\pdfoutline` implementuje záložku. Má tři povinné parametry. Prvním parametrem je klíč, který má stejný význam jako u hypertextového odkazu. Absolutní hodnota druhého parametru specifikuje počet přímých potomků záložky a znaménko specifikuje, zda je záložka otevřena nebo uzavřena (přičemž kladná hodnota znamená otevřenost záložky). Třetím parametrem je vlastní text záložky. Záložky musí být uvedeny v sekvenčním pořadí.
- `\pdfdestxyz`, `\pdfdestfit`, `\pdfdestfith` a `\pdfdestfitv` implementuje cíl pro hypertextový odkaz nebo záložku. Všechny tyto primitivy mají jeden povinný parametr, který specifikuje klíč cíle. Kromě toho má primitiv `\pdfdestxyz` navíc nepovinný parametr, který u tohoto cíle specifikuje zvětšovací faktor. Ostatní údaje pro generování cíle se zjistí implicitně z pozice výskytu primitiv v dokumentu.
- `\pdfdestfitr` a `\pdfendfitr` také implementují cíl pro hypertextový odkaz a záložku. Výskyt `\pdfdestfitr` označuje levý horní roh obdélníku cíle, výskyt `\pdfendfitr` označuje pravý dolní roh. `\pdfdestfitr` má také jeden povinný parametr specifikující klíč cíle.

Primitivy pro specifikace cíle pro hypertextový odkaz a záložku jsou podobné. Tyto primitivy lze implementovat dohromady jako jeden primitiv s jedním parametrem navíc, který specifikuje typ cíle. Důvod, proč byly implementovány zvlášť, je ten, aby syntaxe primitivů byly co nejjednodušší.

Další možná rozšíření

Program `tex2pdf` zatím ještě nevyužívá všechny možnosti, které poskytují TEX a PDF. Proto uvedeme některé oblasti, kde lze program rozšířit o další varianty.

Fonty

Správa fontů programu zatím vyhovuje jen základním požadavkům, aby byl program použitelný. Dalším zlepšením správy fontů může být použití TrueType a METAFONT fontů, případně definování virtuálních fontů pomocí Type3 fontu v PDF.

- Rozsáhlé množství existujících volně šiřitelných TrueType fontů zatím nelze volně použít v TEXu, což je velká škoda. Použití TrueType fontů v TEXu je tak omezené, protože dosud neexistuje žádný nekomerční program, který by generoval `TFM` metriky z TrueType fontů. Také neexistuje žádný nekomerční `DVI` ovladač, který podporuje TrueType fonty. Nyní hlavním úkolem, abychom prosadili použití TrueType fontů v TEXu, je implementace programu `ttf2tfm`, který vytvoří `TFM` metriky pro TrueType fonty.
- Většina fontů používaných v TEXu jsou vytvořeny METAFONTem. Abychom mohli použít METAFONTové fonty i v PDF, nezbývá nejspíš nic jiného než konvertovat je do nějakého formátu, který je podporován v PDF. Nejvhodnější je Type1 formát. Konvertovat METAFONTový formát do Type1 fontů je však problematické, neboť koncept per v METAFONTu je obecnější a nemá přímý odpovídající ekvivalent v jazyce PostScript. Existuje balík `BaKoMa`, který obsahuje Computer Modern fonty v Type1 formátu. Tyto fonty byly podle autora konvertovány z METAFONTu do Type1 formátu plně automatickým způsobem. Pomocí mechanismu Type3 fontu v PDF lze použít `MF` fonty jako bitmapy, ale výsledek je natolik neefektivní, že je lepší použít PostScriptu místo PDF (zobrazení dokumentu s bitmapovými fonty v PDF je *mnohonásobně* pomalejší než v PostScriptu, dokument taky vypadá *mnohem* hůř).
- Myšlenka definování virtuálních fontů pomocí Type3 fontu v PDF byla navržena doktorem Petrem Sojkou jako další možná optimalizace PDF popisu stránky. Tato myšlenka zatím zůstává neimplementována z časových důvodů. Může velmi efektivně redukovat

velikost dokumentů, které obsahují mnoho kompozitních znaků ve virtuálních fontech.

Komprese

Komprese je silná stránka PDF, která je zatím využita jen částečně. Ve vývojové fázi tento nedostatek v podstatě nevadí, ale pro použití v praxi má komprese dat dost velký význam, především v oblasti výměny elektronických dokumentů na počítačové síti, kde je potřeba interaktivně prohlížet dokumenty po síti.

`tex2pdf` podporuje kompresi textu pomocí algoritmu LZW. Předběžné pokusy dávají na textových souborech bez obrázků lepší výsledky než při použití Distilleru na výstup programu `dvips`.

Obrazy, barvy a grafika

Obrazy, barvy a grafika sice nemají mnoho společného s TEXem, ale jsou nutné při sazbě některých dokumentů. Navíc jsou tyto prvky velice dobře podporovány v PDF. Prozatím `tex2pdf` dovoluje pouze vložit přímo do popisu stránky sekvence, které jsou PDF značkovací operátory. To má pochopitelně velice omezenou použitelnost. Abychom mohli využít zmíněné možnosti, potřebujeme mnohem složitější mechanismus, který je nad mé síly.

Zařazení do ε-TEX u

V budoucnosti může být tato práce podnětem pro další bádání, případně by mohla být zařazena do ε-TEXu. ε-TEX je výsledkem práce projektu $\mathcal{NTS}$ (New Typesetting System). Jedná se o rozšíření TEXu na základě zdrojového textu původního programu. Zařazení `tex2pdf` do ε-TEXu by nemuselo být příliš komplikované vzhledem k tomu, že oba jsou implementovány s respektováním způsobu, kterým se TEX doporučuje modifikovat – formou změnových souborů.

Závěr

Program `tex2pdf` je tedy funkční prototyp, který ukazuje možnosti, které poskytují TEX a PDF. `tex2pdf` jako nová technologie umožňuje využívat dobře dokumentované a vysoce stabilní programy, které nejsou

komerční povahy. Poskytuje nyní možnost vytvářet kvalitně formátované dokumenty, které jsou absolutně přenositelné a velmi vhodné pro účel výměny elektronických dokumentů, zvlášť na počítačové síti Internet. `tex2pdf` tak přináší nový postup tvorby dokumentů, který má praktický význam především v oblasti vědecké publikace a přenosu dokomentů.

Program byl prezentován autorovi TEXu, profesoru Stanfordské univerzity Donaldu Knuthovi, a byl jím hodnocen pozitivně. Potřebné změnové soubory pro přeložení programu jsou dostupné na archivu sdružení CSTUG v adresáři:

`ftp://ftp.cstug.cz/pub/tex/local/cstug/thanh/tex2pdf`.

Odkazy

[1] Adobe Systems Incorporated. Adobe Type 1 Font Format—Version 1.1. Addison-Wesley, Reading, MA, USA, August 1990. ISBN 0-201-57044-0.

[2] Adobe Systems Incorporated. Portable Document Format Reference Manual. Addison-Wesley, Reading, MA, USA, 1993. ISBN 0-201-62628-4.

[3] Adobe Systems Incorporated. PostScript Language Reference Manual. Addison-Wesley, Reading, MA, USA, second edition, 1990. ISBN 0-201-18127-4.

[4] Donald E. Knuth. TEX: The Program, Volume B of Computers and Typesetting. Addison-Wesley, Reading, MA, USA, 1986. ISBN 0-201-13437-3.

[5] Han The Thanh. Přenositelný formát dokumentu a sázecí systém TEX. Diplomová práce. FI MU, Brno, 1996. 48 stran.

Han The Thanh
thanh@informatics.muni.cz

Čárové kódy EAN v TEXu

Petr Olšák

Z originálu napsaného v kostrbaté angličtině (článek určený pro TUGboat) pokud možno věrně přeložil Petr Olšák. Důsledkem je poněkud neohrabaná čeština. Věřím, že text přitom neztratil na srozumitelnosti a že i přes tyto „jazykové problémy“ bude pro čtenáře užitečný.

Abstrakt

V tomto článku popíšeme algoritmus na transformaci kódu EAN 13 ve tvaru třináctimístného čísla do čárového kódu (řady různě velkých čar a mezer). Ukážeme implementaci tohoto algoritmu v makrojazyku TEXu. Kreslení čar je realizováno pomocí TEXového primitivu `\vrule`. Dále jsou ukázány některé údaje z norem (například o tolerancích). Makro diskutované níže je k dispozici na CTANu.

Když jsem připravoval svou první knihu o TEXu [3], můj zájem o přípravu této knihy nekončil odevzdáním rukopisu nebo předlohy se sazbou v nějakém nakladatelství. Protože nakladatelem je CSTUG, stal jsem se prakticky sám sobě redaktorem. Dělal jsem si návrh obálky, řešil jsem problémy následné distribuce, získání ISBN atd.

Jakmile jsem získal ISBN (International Standard Book Number), konvertoval jsem toto číslo do EAN 13 (European Article Numbering) a začal se starat o to, jak získat k tomuto číslu čárový kód, protože tento kód bývá obvykle na zadních stranách knižních obálek. Shledal jsem, že ceny pořízení čárového kódu u komerčních firem jsou velmi vysoké. Na druhé straně vytvoření čárového kódu TEXem je velmi přirozená aplikace tohoto programu díky jeho vysoké přesnosti a algoritmickému makrojazyku. Jediným problémem bylo sehnat popis algoritmu, který třináctimístná čísla na vstupu převádí na metriku čar a mezer na výstupu. Tento algoritmus jsem nakonec našel v [1].

Transformace z ISBN do EAN je jednoduchá. ISBN je desetimístné číslo. Pomlčky uvnitř čísla je rozdělují na pole (země-nakladatel-číslo-kontrola) a můžeme tyto pomlčky ignorovat. Na začátek čísla přidáme novou konstatní trojici číslic (978). Dále odebereme poslední kontrolní číslici a vypočítáme ji znovu. Algoritmus na výpočet kontrolní číslice pro ISBN je odlišný od algoritmu používaného u EAN. Pro EAN nejprve spočítáme součet číslic na sudých pozicích. Označíme jej e. Pak spočítáme součet číslic na lichých pozicích (bez kontrolní číslice). Takový součet označíme o. Dále vypočteme $3 \times e + o$. Rozdíl tohoto výsledku od nejbližšího většího násobku deseti je kontrolní číslice. Například TEXbook s pevnou obálkou [2] má ISBN 0-201-13447-0 (0: země USA, 201: nakladatel Addison Wesley, 13447: číslo knihy přidělené nakladatelem, 0: kontrolní číslice). Po přidání tří konstatních číslic na začátek a odstranění kontrolní číslice dostaneme „978020113447?“. Nyní $e = 7 + 0 + 0 + 1 + 4 + 7 = 19$ a $o = 9 + 8 + 2 + 1 + 3 + 4 = 27$. Rozdíl mezi $3 \times 19 + 27 = 84$ a 90 je 6, což je kontrolní číslice. Výsledné EAN můžeme rozdělit pomocí pomlček na pole po šesti číslicích (jen z důvodu snadnějšího čtení) a dostáváme 9-780201-134476.

Transformace z čísla EAN do čárového kódu je poněkud komplikovanější. První číslice je pro knihy 9, nicméně pro ostatní druhy zboží může být jiná. Tato číslice nemá své vlastní pole v čárovém kódu, ale ovlivní algoritmus transformace ostatních číslic do svých polí v čárovém kódu. Šířka čáry nebo mezera mezi čarami v kódu je vyjádřena v násobcích základní jednotky zvané „modul X“. Velikost tohoto modulu může být podle různých předpisů v normě různá (viz SC předpisy níže), ale základní rozměr je 0,33 mm. Každá číslice z pozic 12 až 1 je transformována do pole o šířce 7X (sedm modulů). Toto pole obsahuje vždy dvě čáry a dvě mezery. Dále je na začátek kódu připojen tzv. „startovní znak“ o šířce 3X (1X čára, 1X mezera a 1X čára). Stejný znak je připojen jako tzv. „koncový znak“ na konec kódu a doprostřed kódu mezi 7. a 6. číslici je vložen tzv. „oddělovací znak“ o šířce 5X (1X mezera, 1X čára, 1X mezera, 1X čára a 1X mezera). Tyto speciální znaky mají čáry poněkud delší a přesahují přes dolní okraj o 5X.

Snadno zjistíme, že celková délka EAN kódu je 95X. Dále musíme připočíst bílé místo o šířce 11X vlevo od kódu a 7X vpravo od kódu. Toto je minimální bílý okraj, který je nutno dodržet pro kódy umístěné na barevném pozadí. Celkový počet čar v kódu je 30.

	tab. A	tab. B	tab. C
0	0001101	0100111	1110010
1	0011001	0110011	1100110
2	0010011	0011011	1101100
3	0111101	0100001	1000010
4	0100011	0011101	1011100
5	0110001	0111001	1010000
6	0101111	0000101	1010000
7	0111011	0010001	1000100
8	0110111	0001001	1001000
9	0001011	0010111	1110100

Tabulky A, B a C

Každá číslice je transformována do dvou čar v poli o šířce 7X pomocí jedné ze tří následujících tabulek (A, B a C).

Nula v tabulce znamená bílý modul (šířky X) a jednička označuje černý modul. Například číslice 4 je konvertována podle tabulky A na řadu 1X mezera, 1X čára, 3X mezera a 2X čára a podle tabulky B na řadu 2X mezera, 3X čára, 1X mezera a 1X čára. Všimněme si, že skutečně všechny tabulky konvertují číslice na právě dvě čáry a dvě mezery a že příslušné pole začíná v případě tabulek A a B mezerou a v případě tabulky C čarou.

Číslice z pozic 6 až 1 se tranformují vždy podle tabulky C. Transformace číslic na pozicích 12 až 7 se provede podle tabulky A nebo B. Výběr je podmíněn hodnotou třináctého číslice (první zleva) podle tabulky „Závislost na třináctém číslici“.

Například, pokud je na třináctém pozici číslice 9 (náš případ s knihami) pak jsou podle tabulky A transformovány číslice z pozic 12, 9 a 7 a podle tabulky B číslice z pozic 11, 10 a 8.

Nyní ukážeme TEXovské makro. Nejprve zavedeme speciální font OCRb pro zapsání EAN v čitelné podobě pod čárovým kódem. METAFONTové zdroje těchto fontů jsou k dispozici na CTANu. Vytvořil je Norbert Schwarz. Udělal jsem v těchto zdrojích jeden drobný zásah – přidal jsem na začátek souboru `ocrbmac.mf` příkaz „`mode_setup;`“.

```
\message{The EAN-13 barcodes macro. Copyright (C) Petr Olsak, 1995}
\font\ocrb=ocrb9       % for EAN in ``number form''
\font\ocrbsmall=ocrb7  % for ISBN
```

13. číslice	12	11	10	9	8	7
0	A	A	A	A	A	A
1	A	A	B	A	B	B
2	A	A	B	B	A	B
3	A	A	B	B	B	A
4	A	B	A	A	B	B
5	A	B	B	A	A	B
6	A	B	B	B	A	A
7	A	B	A	B	A	B
8	A	B	A	B	B	A
9	A	B	B	A	B	A

Závislost na třinácté číslici

Nyní deklarujeme některé „proměnné“:

```
\newcount\numlines \newcount\nummodules  % number of bars and of modules.
\newcount\numdigit \newcount\evensum \newcount\oddsum  % internal variables
\newdimen\X            % the module size X,
\newdimen\bcorr        % the bar correction (see bellow).
\newdimen\workdimen \newdimen\barheight   % internal variables
```

Hlavní makro `\EAN` konvertuje třináctimístné číslo EAN na vnitřní reprezanci kódu jako šedesátimístné číslo `\internalcode`. Každá číslice z `\internalcode` reprezentuje násobek modulu X pro čáru nebo mezeru. Číslice jsou napsány ve stejném pořadí, jako je pořadí mezer a čar v kódu. Liché pozice (počítáno zleva) v `\internalcode` reprezentují mezery a sudé čáry.

Makra použijeme například takto: `\EAN 9-780201-134476`. Přítomnost znaků „`-`“ nemá vliv. Makro čte 13 číslic a uloží je do `\firstdigit`, `\frontdigits` a `\enddigits`. Současně makro konvertuje vstup do `\internalcode` použitím `\settables`, `\usetabAB`, `\insertseparator` a `\usetabC`.

```
\def\internalcode{0111}             % Begin mark at start
\def\frontdigits{}                  % 12--7 digit of EAN
\def\EAN{\begingroup\EANscan}
\def\EANscan#1{\if#1-\let\next=\EANscan \else
    \advance\numdigit by1
    \ifnum\numdigit<13
      \ifodd\numdigit \advance\oddsum by #1 \else \advance\evensum by #1 \fi
      \let\next=\EANscan
      \ifnum\numdigit=1 \settables#1\def\firstdigit{#1}\else
      \ifnum\numdigit<8 \usetabAB#1\edef\frontdigits{\frontdigits#1}\else
      \ifnum\numdigit=8 \insertseparator \A \usetabC #1\def\enddigits{#1}%
```

```
      \else             \usetabC#1\edef\enddigits{\enddigits#1}%
      \fi\fi\fi
    \else \testchecksum#1\usetabC#1\edef\enddigits{\enddigits#1}%
      \let\next=\EANclose
    \fi\fi \next}
```

Makro `\testchecksum` kontroluje správnost poslední (kontrolní) číslice kódu EAN.

```
\def\testchecksum#1{\multiply\evensum by3 \advance\evensum by\oddsum
    \oddsum=\evensum
    \divide\oddsum by10 \multiply\oddsum by10 \advance\oddsum by10
    \advance\oddsum by-\evensum \ifnum\oddsum=10 \oddsum=0 \fi
    \ifnum#1=\oddsum \else
    \errmessage{The checksum digit has to be \the\oddsum, no #1 !}\fi}
```

Při expanzi makra `\EANclose` se uzavře `\internalcode` pomocí `\insertendmark`. Dále se napíše do `log` souboru přečtené číslo EAN v třináctimístné formě a též v jeho šedesátimístné vnitřní reprezentaci. Poslední aktivitou je „spuštění“ makra `\EANbox`, které vytvoří box s čarami a mezerami. Vstupním parametrem pro toto makro je šedesátimístné `\internalcode`.

```
\def\EANclose{\insertendmark
    \wlog{EAN: \firstdigit\space\frontdigits\space\enddigits}%
    \wlog{EANinternal: \internalcode}%
    \expandafter\EANbox\internalcode..\endgroup}
```

Jak je vytvářena vnitřní reprezentace `\internalcode`? Na tuto otázku dávají odpověď následující makra. V těchto makrech poznáváme tabulky zmíněné výše, které jsou přepsány do makrojazyka TEXu.

```
\def\A{\def\0{3211}\def\1{2221}\def\2{2122}\def\3{1411}\def\4{1132}%
  \def\5{1231}\def\6{1114}\def\7{1312}\def\8{1213}\def\9{3112}}
\def\B{\def\0{1123}\def\1{1222}\def\2{2212}\def\3{1141}\def\4{2311}%
  \def\5{1321}\def\6{4111}\def\7{2131}\def\8{3121}\def\9{2113}}
\def\settables#1{\ifnum#1=0 \def\tabs{\A\A\A\A\A\A}\fi
                 \ifnum#1=1 \def\tabs{\A\A\B\A\B\B}\fi
                 \ifnum#1=2 \def\tabs{\A\A\B\B\A\B}\fi
                 \ifnum#1=3 \def\tabs{\A\A\B\B\B\A}\fi
                 \ifnum#1=4 \def\tabs{\A\B\A\A\B\B}\fi
                 \ifnum#1=5 \def\tabs{\A\B\B\A\A\B}\fi
                 \ifnum#1=6 \def\tabs{\A\B\B\B\A\A}\fi
                 \ifnum#1=7 \def\tabs{\A\B\A\B\A\B}\fi
                 \ifnum#1=8 \def\tabs{\A\B\A\B\B\A}\fi
                 \ifnum#1=9 \def\tabs{\A\B\B\A\B\A}\fi}
\def\usetabAB#1{\expandafter\scantab\tabs\end \usetabC #1}
\def\scantab#1#2\end{#1\def\tabs{#2}} % The tab #1 is activated and removed
\def\usetabC#1{\edef\internalcode{\internalcode\csname#1\endcsname}}
\def\insertseparator{\edef\internalcode{\internalcode 11111}}
\def\insertendmark{\edef\internalcode{\internalcode 111}}
```

Není potřeba definovat tabulku C, protože tato tabulka je přesně „negativní“ k tabulce A. Vložením oddělovacího znaku (řádek 52 v ukázce) se přidá do `\internalcode` lichý počet číslic (jmenovitě 5). Z toho plyne, že se převrátí význam mezery a čáry pro další číslice. Použití tabulky `\A` je pak pro další transformace číslic z pozic 6 až 1 dostačující.

Nyní přichází nejdůležitější část našeho makra – vytváření čar pomocí primitivu `\vrule`. Tuto práci dělá interní makro `\EANbox`. Makro postupně čte jako parametr šedesáticiferné číslo `\internalcode` ukončené dvěma tečkami. V každém kroku načte dvě číslice (první reprezentuje mezeru a druhé šířku čáry) a vloží odpovídající `\kern` a `\vrule`. Každá dvojice kern/rule je navíc korigována podle tzv. „čárové korekce“. Norma doporučuje udělat každou čáru poněkud užší, než vyplývá z přesného násobku modulu X. Toto doporučení vyplývá z vlastností inkoustu při tisku. Například pro ofsetovou technologii se doporučuje redukovat čáry o 0.020 mm. Při redukci šířek čar samozřejmě musíme zvětšit o stejnou hodnotu mezery, abychom udrželi stejnou vzdálenost mezi čarami.

Čáry 1, 2, 15, 16, 29 a 30 mají nenulovou „hloubku“ (5X), protože se jedná o čáry startovního, oddělovacího a koncového znaku. Za normálních okolností je výška čáry 69,24 X, ale tato výška může být redukována třeba v případě, že je nahoru ke kódu přidáno číslo ISBN. Jestliže je `\barheight` nulový, pak je použita implicitní výška. Jinak je použita výška čáry podle `\barheight`. Tato vlastnost dává uživateli možnost nastavit výšku čáry individuálně.

```
54 \def\EANbox{\vbox\bgroup\offinterlineskip
55   \setbox0=\hbox\bgroup \kern11\X\EANrepeat}
56 \def\EANrepeat#1#2{\if#1.\let\next=\EANfinal \else\let\next=\EANrepeat
57   \advance\numlines by1
58   \advance\nummodules by#1 \advance\nummodules by#2
59   \workdimen=#1\X \advance\workdimen by \bcorr \kern\workdimen
60   \workdimen=#2\X \advance\workdimen by-\bcorr \vrule width\workdimen
61      \ifdim\barheight=0pt height 69.24242424\X \else height\barheight \fi
62      \ifnum\numlines=1    depth5\X\else  % the start mark
63      \ifnum\numlines=2    depth5\X\else
64      \ifnum\numlines=15   depth5\X\else  % the separator mark
65      \ifnum\numlines=16   depth5\X\else
66      \ifnum\numlines=29   depth5\X\else  % the end mark
67      \ifnum\numlines=30   depth5\X\else depth0pt \fi\fi\fi\fi\fi\fi
68   \fi\next}
```

Makro `\EANfinal` kontroluje správnost přečteného `\internalcode`. Počet číslic musí být 60 a součet číslic musí být 95 (protože je 95 modulů X v kódu celkem). Jestliže tyto podmínky nejsou splněny, je aktivováno makro `\internalerr`. Tato situace by neměla nikdy nastat. Tato

chyba signalizuje, že je chybná nějaká interní tabulka nebo je porušena konzistence makra.

Makro `\EANfinal` sestaví `\vbox`. „Hloubka“ vnitřního `\hbox`u s kódem je 5X, protože taková je „hloubka“ čar pro startovní, oddělovací a koncový znak. Přepíšeme tuto hloubku na 0pt, abychom mohli přidat číslo EAN v čitelném tvaru těsně pod kód.

```
\def\EANfinal{\testconsistence
  \kern7\X\egroup
  \hbox{\ocrbsmall \kern10\X \ISBNnum}\kern1\X
  \dp0=0pt \box0 \kern-1\X
  \hbox{\ocrb\kern2\X\firstdigit\kern5\X \frontdigits\kern5\X \enddigits}
  \egroup \global\barheight=0pt \gdef\ISBNnum{}}
\def\testconsistence{\ifnum\numlines=30\else\internalerr\fi
  \ifnum\nummodules=95\else\internalerr\fi}
\def\internalerr{\errmassage{Sorry, my internal tables are wrong, may be.}}
```

Jestliže uživatel napíše ISBN pomocí makra `\ISBN` (například `\ISBN 0-201-13447-0`), tento údaj se připojí nad kód a výška čar v kódu bude redukována.

```
\barheight=0pt
\def\ISBNnum{}
\def\ISBN #1 {\def\ISBNnum{ISBN #1}\barheight=45.151515\X\relax}
```

Konečně definujme velikost modulu `\X` a čárovou redukci.

```
\X=.33mm       % Basic size 100%, SC2 code
\bcorr=.020mm  % Bar-correction for offset process
\endinput
```

Makro je uloženo v souboru `ean13.tex` a bylo testováno v plainu následujícím způsobem:

```
\input ean13
\nopagenumbers
\ISBN 80-901950-0-8  \EAN 978-80-901950-0-4   % Typografický systém TeX
\end
```

Výsledek vypadá takto:

Makro pracuje rovněž v LaTeXu 2.09. LaTeX 2$_\varepsilon$ nebyl testován.[1]

[1] Makro bylo otestováno v redakci. LaTeX 2$_\varepsilon$ lze použít, avšak je nutno mít na

V závěru tohoto článku srovnáme povolené tolerance popsané v normě, přesnost TEXu a možnosti některých výstupních zařízení.

Velikost modulu X může být různá. Makro uvedené výše vytváří čárové kódy EAN pro základní velikost modulu X 0,33 mm. Tato velikost je popsána v tzv. SC2 variantě normy jako základní velikost 100%. Norma umožňuje použít některé další velikosti modulu X. Změnou parametru `\X` v našem makru můžeme získat další velikosti kódu. Samozřejmě, že musíme též změnit velikost zavedeného fontu OCRb.

Možnosti povolených velikostí modulu X jsou popsány v tabulce „Různé velikosti modulu X".

Malé rozměry modulu X jsou doporučeny pro výstupní zařízení s vysokou kvalitou a velké velikosti X dávají možnost kreslit čárové kódy na zařízeních s nižším rozlišením.

Norma popisuje tři speciální toleranční parametry, které jsou závislé na velikosti modulu X. Parametr a specifikuje toleranci pro šířku čáry, parametr b udává toleranci pro vzdálenost mezi hranami (levá nebo pravá hrana ale vždy stejná) dvou sousedních čar a parametr c popisuje toleranci pro šířku pole odpovídající jedné číslici (tedy přesně šířku 7X). Následující tabulka „Tolerance" popisuje tyto tolerance v mikrometrech (μm). Nechápu, proč sloupec „vel. X" z předchozí tabulky přesně neodpovídá témuž sloupci „vel. X" z následující tabulky. Bohužel, normy často bývají tajemné.

Nyní můžeme začít srovnávat. Uvažujme základní velikost kódu 100% (modul X je 0,33 mm). Tolerance šířky čáry je 101 μm, (ne)přesnost TEXu je 0,0054 μm, velikost pixelu pro osvitovou jednotku při 2 400 dpi je zhruba 10 μm a doporučená čárová redukce pro ofsetovou technologii je 20 μm. Pokud nastavíme osvitovou jednotku na 1 200 dpi, je nepřesnost jejího výstupu srovnatelná s doporučenou čárovou redukcí pro ofset.

V závislosti na vnitřním algoritmu `dvi` ovladače se tedy vysoká přesnost TEXu ztrácí a některé toleranční parametry mohou způsobit těžkosti. Zaokrouhlovací algoritmy `dvi` ovladačů mohou pracovat dvěma různými způsoby. Za prvé, ovladač umístí každou jednotlivou čáru (rule) do výstupního rastru individuálně a zaokrouhlí až v okamžiku umístění. Podle druhého způsobu ovladač nejprve zaokrouhlí řadu kern, rule, kern,

paměti, že volba `split` ve stylu `czech` změní znak minus na aktivní. Před použitím uvedených maker je tudíž nezbytné zpětné nastavení kategorie znaku minus na 12. Pozn. red.

vel. X	norma	koef.	rozměr včetně okrajů
0,264	SC0	0,800	29,83 × 21,00
0,270	SC0	0,818	30,58 × 21,53
0,281	SC0	0,850	31,70 × 22,32
0,297	SC1	0,900	33,56 × 23,63
0,313	SC1	0,950	35,43 × 24,94
0,330	SC2	1,000	37,29 × 26,26
0,346	SC2	1,050	39,15 × 27,58
0,363	SC3	1,100	41,02 × 28,29
0,379	SC3	1,150	42,88 × 30,20
0,396	SC4	1,200	44,75 × 31,51
0,412	SC4	1,250	46,61 × 32,82
0,429	SC5	1,300	48,48 × 34,14
0,445	SC5	1,350	50,34 × 35,45
0,462	SC5	1,400	52,21 × 36,76
0,478	SC5	1,450	54,07 × 38,08
0,495	SC6	1,500	55,94 × 39,39
0,511	SC6	1,550	57,80 × 40,70
0,528	SC7	1,600	59,66 × 42,01
0,544	SC7	1,650	61,53 × 43,33
0,561	SC7	1,700	63,39 × 44,64
0,577	SC7	1,750	65,26 × 45,96
0,594	SC8	1,800	67,12 × 47,26
0,610	SC8	1,850	68,99 × 48,58
0,627	SC8	1,900	70,85 × 49,90
0,643	SC8	1,950	72,72 × 51,20
0,660	SC9	2,000	74,58 × 52,52
0,700	SC10	2,120	79,05 × 55,67

Různé velikosti modulu X

rule... na jednotky pixelů a pak už při umisťování čar pracuje jen s těmito jednotkami. Tento přístup může vést k akumulaci chyby a může způsobit potíže s parametrem *c*. Zdá se ale, že pro čtečky čárových kódů je tento druhý přístup mnohem přijatelnější, protože důležitější je dodržet rovnoměrné vzdálenosti mezi sousedními objekty, než dbát na dodržení parametrů pro globální šířku kódu. To je ale jen můj názor.

vel. X	$\pm a$	$\pm b$	$\pm c$
0,26	32	38	75
0,28	52	41	81
0,30	72	44	87
0,32	92	47	93
0,33	101	49	96
0,34	105	50	99
0,36	115	53	104
0,38	124	56	110
0,40	134	59	116
0,42	143	62	122
0,44	152	65	128
0,46	162	68	133
0,48	171	71	139
0,50	181	73	145
0,52	190	76	151
0,54	199	79	157
0,56	209	82	162
0,58	218	85	168
0,60	228	88	174
0,62	237	91	180
0,64	246	94	186

Tolerance

Zjistil jsem, že `dvi` ovladače většinou zaokrouhlují šířku čáry (rule) na jednotky pixelů *nahoru* a nikdy ne *dolů*. Důsledkem toho je skutečnost, že mezery mají tendenci být o jeden pixel užší než šířka čar, které by podle pravidel měly mít stejnou velikost. Proto doporučuji přidat k čárové redukci (k registru `\bcorr`) ještě navíc polovinu velikosti pixelu výstupního zařízení.

Slyšel jsem, že čárové kódy EAN jsou úspěšně čteny i ze štítků vytištěných pomocí jehličkových tiskáren s velmi nízkým rozlišením. Přitom je použit základní rozměr 100% nebo jen nepatrně větší. Ve světle této skutečnosti můžeme prohlásit, že tolerance vstupních zařízení čárových kódů jsou obvykle podstatně větší než tolerance předepsané normou.

Literatura

[1] Adriana Benadiková, Štefan Mada a Stanislav Weinlich. *Čárové kódy, automatická identifikace*. Grada 1994, 272 stran. ISBN 80-85623-66-8.

[2] Donald Knuth. *The TEXbook*, díl A z *Computers & Typesetting*. Addison-Wesley, Reading, MA, USA 1986, ix+483 stran. Pevná obálka. ISBN 0-201-13447-0.

[3] Petr Olšák. *Typografický systém TEX*. CSTUG 1995, 270 stran. ISBN 80-901950-0-8.

LaTeXová kuchařka/1

Zdeněk Wagner

Máte-li přístup na Internet, jistě jste již několikrát zaznamenali diskusi, zda je lepší plain TeX nebo LaTeX. Část této polemiky se dostala i na stránky Zpravodaje č. 4/94. Mezi TeXperty je dost zakořeněn mylný názor, že plain TeX poskytuje principiálně kvalitnější sazbu. Ukážeme velmi snadno, že takový názor správný není.

Na počátku zpracování dokumentu máme ASCII soubor. Ten předložíme programu, v OS/2 a v DOSu je to `tex*.exe`, který si navíc načte tzv. formát, obsahující definice základních maker, a metriky fontů. Při zpracování se makra, použitá v dokumentu, expandují s použitím sekvencí definovaných ve formátu až na TeXovské primitivy. V tomto stadiu není rozhodující, v jakém formátu byla makra definována. Výsledné primitivy jsou vždy stejné, tentýž je i algoritmus řádkového a stránkového zlomu. Typografická kvalita dokumentu je tedy v zásadě nezávislá na volbě formátu.

Při prohlížení vytištěných dokumentů ovšem často dojdete k závěru, že výše uvedené tvrzení neplatí. Příčina zřejmě tkví v nezkušenosti některých uživatelů. Standardní třídy ARTICLE, REPORT a BOOK hýří velikostmi písma pro nadpisy, takže výsledek vypadá dosti humpolácky. V LaTeXu se to snadno podaří každému začátečníkovi, který si přečte, že pro název kapitoly má použít `\chapter`. Plain TeX je zde trochu složitější. Začátečník ještě nezná všechny možnosti, a proto jeho dokumenty

vypadají decentněji. Zůstává zde reálná šance, že bude s kvalitou svých textů spokojen a nebude je v budoucnu proměňovat na vzorníky fontů. Výsledek tedy záleží pouze na tom, zda uživatel dokáže své typografické cítění převést do maker formátu, který si pro svoji práci vybral.

Nyní se již každý může svobodně rozhodnout, který formát bude používat. Zde jsem ve sporu s Karlem Horákem. Preferuji LaTeX, protože vše je v něm již uděláno. Makra pro názvy kapitol zapisují vhodná makra pro tisk obsahu do pomocného souboru, takže pak na začátku či konci dokumentu stačí uvést `\tableofcontents`. Plain TeX nic takového nemá, každý si to ve svém dokumentu musí udělat sám. Je pravda, že život je složitý. Každá kniha má názvy kapitol formátovány jinak, liší se vzhled obsahu, seznamu tabulek, rejstříku, atd. Zdálo by se, že stejně v každé knize musíme vše nadefinovat znovu. Naštěstí jsou všechna standardní makra definována strukturovaně. Pokud uživatel ví, kam zasáhnout, stačí předefinovat poměrně malou část a vlastní mechanismus zůstává beze změny. Vzhled LaTeXového dokumentu se pak změní k nepoznání.

Problém práce s LaTeXem tedy spočívá v tom, že uživatel musí vědět, kam zasáhnout. Tento seriál začal vznikat v mé mysli jako návod pro ty, kdo se standardními LaTeXovými třídami nejsou spokojeni, a nechtějí sami studovat `latex.tex`. Touto cestou jsem se kdysi vydal já, když jsem žádnou literaturu nemohl sehnat. Přestože jsem později svá makra upravil studiem knih, které si mi podařilo koupit, mohou občas být zbytečně těžkopádná.

Počet dílů tohoto seriálu je definován jako n, přičemž hodnota n není určena. První pokračování je již zhruba připraveno, počet a zaměření dalších dílů bude záviset na čtenářském ohlasu.

Většina příkladů, které budou v seriálu použity, vychází z maker skutečně použitých při sazbě knih. Makra ovšem byla vytvořena pro specifické prostředí, takže nemusí plně vyhovovat požadavkům všech uživatelů. Kromě toho je nutné důrazně upozornit, že v některých příkladech je kladen důraz na demonstraci makrojazyka, ale nedbá se vůbec na typografickou kulturu. Než použijete nějaký příklad ve své produkci, zamyslete se, zda příslušné makro typograficky zapadá do vzhledu dokumentu.

V následujícím textu se budu úmyslně dopouštět nepřesností. Podrobný výklad by totiž nebyl dostatečně přehledný a zbytečně by uživatele zatěžoval detaily, které pro běžnou práci nepotřebují vědět. Zájemci se pak mohou dovědět více vlastním studiem různých knih.

1. Proč psát vlastní makra?

V počátečních odstavcích tohoto článku bylo uvedeno, že pro LaTeX je téměř vše hotovo. Když jsem sám s LaTeXem začínal a s něčím jsem si nevěděl rady, dostával jsem odpovědi typu: „Použijte styl ... , který je k dispozici na ....“ Příslušné styly jsem sehnal, otestoval a zjistil, že mi vyhovují. Pak jsem je použil všechny současně, abych ve svém dokumentu získal vše potřebné. Výsledkem byla zpráva:

```
TeX capacity exceeded
```

Je jistě příjemné, když nějakou práci za vás udělá někdo jiný. Nicméně, není vždy nutné chodit na mravence s kanónem. Často místo složitých stylů postačí jednoduché makro. Programátora pak potěší, že dokázal problém vyřešit sám. Navíc nebude zatěžovat paměť svého počítače horou maker, které nehodlá využít.

Musím přiznat, že ve svých začátcích jsem používal AT 286 a DOS. Tím byla dána paměťová omezení. Nyní provozuji TeX v OS/2 na počítači s 32 MB paměti, takže bych se o paměť starat nemusel. Přesto i nyní dávám přednost vlastním jednoduchým makrům před složitými styly.

Neméně podstatnou úlohu totiž hraje (ne)snadnost získávání stylů. Vše je k dispozici na CTAN. Co ale mám dělat, když potřebuji udělat něco odpoledne doma, výsledek musí být odevzdán příští den a z domova se na síť nedostanu. I to je jeden z důvodů, proč vzniká tento seriál.

2. LaTeX 2.09 versus LaTeX 2$_\varepsilon$

Často se uživatelé ptají, proč by měli přejít na LaTeX 2$_\varepsilon$, co jim to přinese nového a jaké budou mít potíže. Pak se potýkají s nedostatkem paměti a uvažují, zda to vůbec mělo smysl.

Řada maker v mé kuchařce pochází z doby, kdy LaTeX 2$_\varepsilon$ ještě neexistoval a fungují s oběma verzemi. Budou zde ovšem presentována i makra, která s verzí 2.09 nechodí.

Pro běžného uživatele nepřináší LaTeX 2$_\varepsilon$ nic převratného. Může dál používat své navyklé způsoby vytváření dokumentů a vše bude fungovat. Výhody ocení zejména programátor maker. Pro uživatele však také poskytuje nové možnosti.

Hlavní novinkou je nová verze New Font Selection Scheme, NFSS2. Máme k dispozici ortogonální systém přepínání jednotlivých atributů

písma, takže `\bfseries` zapne písmo **tučné**, `\itshape` přepne na *kurzívu* a kombinace `\bfseries \itshape` v libovolném pořadí poskytne ***tučnou kurzívu***. Tyto možnosti měla i původní NFSS vytvořená pro LaTeX 2.09. Ta ovšem využívala přepínače `\bf` a `\it`, takže dokumenty pak vypadaly odlišně podle toho, jakým formátem byly zpracovány. Některé dokumenty dokonce zpracovat nešly. NFSS2 tento problém řeší. Původní makra `\bf`, `\it` apod. mají stejné funkce jako v LaTeXu 2.09, nové vlastnosti se dosahují novými makry. Zpracování dokumentu, který byl psán pro starou NFSS, pak vyžaduje speciální styl, kde jsou stará makra předefinována.

Při psaní kurzívou nesmíme zapomenout na kurzívní korekci `\/`. Této starosti se vyhneme použitím nového makra `\textit`. Všechna makra `\text..` totiž automaticky vkládají kurzívní korekce tam, kde jsou potřebné. Uveďme si jako příklad následující větu:

Významné *slovo* sázíme kurzívou.

Tuto větu jsme vysázeli takto:

```
Významné \textit{slovo} sázíme kurzívou.
```

Všimněte si, že kurzívní korekce zde není explicitně zapsána, protože makro `\textit` ji dosadí samo. Výhodou je zdánlivě pouze to, že na kurzívní korekci nemusíme myslet. Podívejme se však znovu na větu a rozhodněme se, že ji trochu doplníme:

Významné *slovo*, které chceme zdůraznit, sázíme kurzívou.

Zde jsem za slovo tištěné kurzívou přidali čárku, takže kurzívní korekce tam být nesmí. Při použití `\it` nebo `\itshape` bychom kurzívní korekci museli odstranit ručně a třeba někdy jindy, až bychom čárku odstranili při další změně formulace, museli znovu dopsat. Makro `\textit` nás této úmorné činnosti ušetří.

Podobnou vlastnost má i makro `\emph`. Můžeme tedy psát:

```
Významné \emph{slovo} sázíme kurzívou.
```

a výsledkem bude: „Významné *slovo* sázíme kurzívou.“ Kurzívní korekce bude opět automaticky doplněna makrem `\emph`.

Pokud sázíme hlavní text kurzívou, pak se pro vyznačování používá písmo stojaté. Nyní musí přijít kurzívní korekce na začátek. I tento případ bude automaticky ošetřen. Stejně zapsaná věta se nyní vytiskne jako: „Významné slovo *sázíme kurzívou.“*

Nové možnosti přinášejí i makra pro definici maker. TEX poskytuje primitiv `\def`. Při jeho použití máme plnou kontrolu nad způsobem, jak bude nové makro definováno. Můžeme bez varování předefinovat již existující makro a dokonce i primitiv. V mnoha případech se to hodí. Když se nám podaří předefinovat nějaké důležité makro omylem, nestačíme se divit. Jako domácí úkol si vyzkoušejte následující definici:

```
\def\output{\typeout{Kontrola}}
```

Výsledek bude otřesný. Při použití definičního makra `\newcommand` by se to nestalo. Proto je `\newcommand` vhodný zejména pro začátečníky. LATEX 2_ε navíc dává uživatelům vylepšenou verzi maker `\newcommand` a `\renewcommand`. Vylepšení spočívá v možnosti specifikace nepovinného parametru, který bude uveden v hranatých závorkách. Definujme si například:

```
\newcommand\pokus[1][slovo]{Tiskneme \textit{#1} kurzívou.}
```

Zápisem `\pokus` pak dostaneme: „Tiskneme *slovo* kurzívou.“ Můžeme též napsat `\pokus[cokoliv]` a výsledkem bude: „Tiskneme *cokoliv* kurzívou.“ Je-li parametrů více, můžeme tento trik použít pouze pro první z nich.

Pokud chceme změnit existující definici, musíme použít jiné makro: `\renewcommand`. Co ale máme dělat v případě, chceme-li definovat makro `\pokus` jen za předpokladu, že ještě definováno není? Pokud použijeme `\newcommand` a makro je již definováno, bude se nová definice ignorovat a LATEX oznámí chybu. Cíle bylo dosaženo, ale s chybovým hlášením. Lze to samozřejmě obejít různě, ale LATEX 2_ε má elegantní řešení. Tím je definiční makro `\providecommand`, které má stejnou syntaxi jako `\newcommand`.

Další vymoženosti, které získáte přechodem na LATEX 2_ε, budou uváděny postupně v dalších kapitolách a v dalších dílech kuchařky. Kdybychom je chtěli popsat na tomto místě, nezbylo by nám, než vměstnat celý seriál do této jediné kapitoly.

3. Nefunguje `\vspace`

Zkušený LATEXista mi snad odpustí, že zde budu vysvětlovat zcela triviální problém. Na otázku, proč `\vspace` vkládá mezeru jinam, než má,

jsem už odpovídal tolikrát, že to stojí za popsání v tomto seriálu.

Nejprve si musíme říci něco o tom, jak TEX zpracovává dokument. Velmi zjednodušeně lze říci, že TEX pracuje ve dvou režimech[1], vertikálním a horizontálním. Na začátku dokumentu je TEX přepnut do vertikálního režimu. Zde se může vyskytovat jakýkoliv vertikální materiál, tj. cokoliv, co se sází ve svislém směru. Nelze sem tedy vkládat žádný text. Jakmile se v dokumentu objeví text nebo jiná horizontální položka, přepne se TEX do horizontálního režimu. Celý text včetně dalších horizontálních položek se načítá až do konce odstavce. Ten se pak zlomí na řádky, které se vloží do hlavního vertikálního seznamu. Potom se TEX přepne do vertikálního režimu.

Je zřejmé, že vertikální mezery nelze vkládat uvnitř odstavce. TEX proto při objevení primitivu `\vskip` ukončí odstavec a potom vloží vertikální mezeru. Někdy ovšem potřebujeme vložit vertikální položku (penaltu nebo svislou mezeru) mezi řádky odstavce. K tomu slouží primitiv `\vadjust`.

LATEXovské makro `\vspace` slouží pro oba účely. Pokud se vyskytuje ve vertikálním režimu, tedy mezi odstavci, expanduje se na `\vskip`. Jako parametr makra lze zadat „gumovou“ mezeru. Příkaz:

```
\vspace{4cm plus 2cm minus 1cm}
```

je ekvivalentní primitivu

```
\vskip 4cm plus 2cm minus 1cm
```

Příkaz vloží mezeru výšky 4 cm. Mezeru lze rozpálit na 6 cm nebo stáhnout na 3 cm.

V horizontálním režimu vkládá `\vspace` mezeru prostřednictvím primitivu `\vadjust`. Pro ilustraci je těsně před tečku, která ukončuje tuto větu, vložen příkaz `\vspace{11dd}`. Všimněte si, že mezera se vloží *za* řádek, na němž je příkaz uveden. Pokud vložíte `\vspace` na nevhodné místo uvnitř odstavce, může TEX udělat mezeru jinde, než bylo zamýšleno. To se stává zejména tehdy, když použijete `\vspace` uprostřed makra. Často se takto vložený `\vspace` dostane na začátek odstavce, takže místo mezery mezi odstavci dostanete mezeru za prvním řádkem odstavce.

[1] Podrobnější výklad viz The TEXbook, kapitola 13.

Podobně se chovají i makra `\smallskip`, `\medskip` a `\bigskip`. Ta byla totiž předefinována pomocí `\vspace` a platí o nich totéž, co bylo uvedeno výše.

Vertikální mezery se automaticky zruší na začátku stránky. Pokud skutečně chcete začít stránku svislou mezerou, musíte za `\vspace` přidat hvězdičku.

4. Invalid use of `\spacefactor...`

Toto je běžná chybová zpráva, se kterou se potýká každý začínající LaTeXista a neustále se příslušný dotaz objevuje i v diskusních listech. Pochopení této chyby je nezbytné dříve, než se pustíme do dalších výkladů.

V TeXu má každý znak dva atributy, kód a kategorii. Kategorie znaků jsou shrnuty v tabulce 1. Některé názvy jsou ponechány v angličtině, protože český překlad zřejmě není zaveden.

Určité znaky mají kategorie přiděleny automaticky, některé kategorie jsou předefinovány ve formátu. Primitiv `\catcode` umožňuje změnu kategorie kdekoliv v textu. Tímto mechanismem byl v tabulce vytištěn znak `^`. Změnou kategorie na 12 ztratil svůj speciální význam.

Nebudeme se zde příliš zabývat vysvětlováním kategorií. Připomeneme, že aktivní znak se chová jako makro. Můžeme tedy aktivovat například znak „č“ a definovat jeho význam:

```
\def č{\v{c}}
```

Každý výskyt „č“ v textu pak bude nahrazen příkazem `\v{c}`.

Mnohem důležitější jsou pro nás kategorie 11 a 12. Víme, že makro je uvedeno speciálním „escape“ znakem, za nímž následuje posloupnost písmen nebo jeden nepísmenový znak. První případ jsme již viděli třeba v makru `\vspace`, druhým případem je zúžená mezera `\,`.

Formát i styly obsahují vnitřní makra, která by se v textu dokumentu užívat neměla. Je to zařízeno jednoduchým trikem. Kategorie znaku „@“ je změněna na 11. TeX potom považuje „@“ za písmeno, takže můžete definovat i používat makro `\@startsection`. Ve vlastním dokumentu má „@“ kategorii 12. Pokud tedy napíšete `\@startsection`, bude to TeX interpretovat jako makro `\@` následované znaky `startsection`. `\@` se

Tabulka 1: Kategorie znaků

Kat.	`Příklad`	Význam
0	`\`	Escape
1	`{`	Začátek skupiny
2	`}`	Konec skupiny
3	`$`	Přepínač matematického režimu
4	`&`	Tabelátor
5	`^^M`	Konec řádku
6	`#`	Parametr
7	`^`	Exponent
8	`_`	Index
9		Ignorovaný znak
10		Mezera
11	`a`	Písmeno
12	`1`	Jiný znak
13	`č`	Aktivní znak
14	`%`	Komentář
15		Neplatný znak

expanduje na `\spacefactor` a tento primitiv má smysl jen za určitých okolností.

Obvykle se začátečník dozví v odpovědi na svoji otázku, že má předefinovat nějaké makro. Vloží tedy do preambule kód obsahující například `\@startsection`. Zde se ovšem `\spacefactor` vyskytovat nesmí, což způsobí chybu uvedenou v nadpisu. V preambuli se nesmí vyskytovat žádný text, proto LaTeX oznámí:

```
Missing \begin{document}
```

„Chybějící“ `\begin{document}` je přidán a vytištěn text `startsection`. Pokud se dále vyskytují makra, která jsou povolena pouze v preambuli, dostanete další chybové zprávy, přinejmenším na `\begin{document}`.

Makra obsahující „@“ lze tedy definovat a používat pouze tam, kde je „@“ písmenem. Je tomu tak v souborech s příponou `.sty`, které jsou načítány příkazem `\usepackage`. Chcete-li taková makra používat v definicích intenzivně, je vhodné, abyste si napsali vlastní soubor s makry a v preambuli jej načetli makrem `\usepackage`. Pokud potřebujete

v preambuli jen jednu definici, můžete použít `\makeatletter` pro změnu kategorie znaku „@“ na 11. Zpětnou změnu kategorie pak provedete makrem `\makeatother`.

První z uvedených metod je využita i při tvorbě tohoto zpravodaje. Všechny základní definice jsou v CSBUL.STY a hlavní TEXový soubor začíná příkazy:

```
\documentclass[twoside]{article}
\usepackage{csbul}
```

5. Vizuální a kontextové značkování

V předchozích kapitolách jsme si naznačili základní problémy, se kterými se potká snad každý začátečník. Nyní již můžeme rozebírat složitější případy. Před vlastním popisem maker se ovšem ještě zastavíme trochu u filozofie LATEXu.

Všichni víme, že zdrojový dokument pro TEX je obyčejný ASCII text. Vzhled dokumentu je popsán řídícími příkazy. Tomuto způsobu popisu stránky říkáme značkování (anglicky markup).

Jestliže chceme napsat slovo **tučně**, použijeme buď zápis {`\bfseries tučně`} nebo `\textbf{tučně}`. V obou případech definujeme, jak se má uvedené slovo vytisknout. Použitý příkaz určuje vizuální vzhled a odpovídá zvolení stylu *tučně* ve WYSIWYG editoru. Tomuto způsobu označování vzhledu části textu říkáme *vizuální značkování*.

Vizuální značky mají svůj pevný význam bez ohledu na to, kde se vyskytují. Objevíme-li při čtení zdrojového textu `\textbf`, víme, že argument makra bude vytištěn tučně. Tento přístup má tedy výhodu v tom, že je zde jasně patrná korespondence mezi příkazem a vzhledem části textu.

Nyní si představme následující situaci. Píšeme článek do sborníku, kde názvy kapitol mají být tučně. Použijeme pro ně `\textbf`. V textu chceme zdůraznit některá slova. Opět použijeme `\textbf`, abychom je vysázeli tučně. Článek odevzdáme a redaktor nám oznámí, že zvýrazněná slova nemají být tučná, ale v kurzívě. Musíme tedy některá `\textbf` přepsat na `\textit`. Ještě horší to může být s nadpisy. Obvykle se nadpis odděluje od předchozího i následujícího textu mezerou. V našem článku to mohlo vypadat například takto:

```
\vspace{22pt plus 11pt minus 6pt}
\noindent\textbf{Nadpis}

\vspace{11pt plus 6pt minus 3pt}
\noindent Začátek odstavce...
```

Nyní si představte, že redator bude vyžadovat jiné mezery nad a pod nadpisem. V takovém případě nezbyde než projít ručně celý dokument a provést mnoho úprav, které jsou nudné a pracné a navíc se při tom dá napáchat spousta chyb. Zde vizuální značkování předvádí názorně své nevýhody.

Problém se snadno vyřeší *kontextovým značkováním*. Základní rozdíl spočívá v tom, že kontextová značka definuje *logický význam* části textu, nikoliv jeho vizuální podobu. Za nejjednodušší kontextovou značku bychom snad mohli považovat `\emph`, protože LaTeXu říká, že příslušná pasáž má být zvýrazněna. Konkrétní způsob zvýraznění záleží na tom, jakým písmem je sázen základní text.

Nesporně kontextovými značkami jsou makra pro sazbu názvů kapitol, podkapitol, apod. Makro `\section` provádí řadu činností. Nejenže vytiskne název kapitoly odpovídajícím stylem, ale navíc může automaticky kapitolu očíslovat, připraví makra pro křížové odkazy, předá informace pro živé záhlaví, zabrání stránkovému zlomu pod názvem kapitoly i za prvním řádkem prvního odstavce, vytvoří záznam pro tisk obsahu. Bylo by velmi pracné a nepohodlné, kdybychom toto vše museli mnohonásobně ručně zapisovat do zdrojového textu.

V konečné fázi nás vždycky zajímá vizuální vzhled. Kontextová značka tedy musí být definována pomocí vizuálních značek. Tento přístup má výhodu zejména v tom, že vzhled všech objektů stejného logického významu snadno udržíme jednotný a v případě potřeby jej můžeme změnit předefinováním jediného makra.

V předchozích příkladech jsme použili `\textbf` jak pro názvy kapitol, tak pro zvýraznění slov v textu. Při čtení zdrojového textu pak snadno zjistíme, která část bude vytištěna tučně. Nemusí však být na první pohled jasné, zda se jedná o název kapitoly či zvýrazněný text.

Při kontextovém značkování použijeme `\section` pro nadpisy a `\duraz` pro zvýrazněný text. Chceme-li zvýrazňovat tučně, použijeme definici:

```
\let\duraz\textbf
```

Po intervenci z redakce, požadující zvýrazňování kurzívou, změníme tuto definici na:

```
\let\duraz\textit
```

nebo:

```
\let\duraz\emph
```

Příklad najdete i v tomto zpravodaji. Ve všech článcích jsou názvy kapitol označeny makrem `\section`. Vizuálně vypadají stejně, pouze v některých článcích jsou kapitoly číslovány. Je toho dosaženo jednoduchou změnou definice, o níž si povíme později.

Podobně je to i s názvem článku. Zdrojový text tohoto příspěvku začíná (zjednodušeně) příkazy:

```
\begin{clanek}
\title{\LaTeX ová kuchařka/1}
\author{Zdeněk Wagner}
\podpis{Zdeněk Wagner\\\texttt{wagner@mbox.cesnet.cz}}
\maketitle
```

Vidíte, že jsou zde použita makra `\title`, `\author` a `\maketitle`. Vizuálně však produkují něco jiného než standardní třída ARTICLE.

Nyní je zřejmá další výhoda kontextového značkování. To nám totiž umožňuje sjednotit způsob označování konkrétních logických objektů. Autor pak píše stejným způsobem článek do časopisu, do sborníku konference i do tohoto zpravodaje.

Představte si, že máte vysázet knihu a rukopis dostanete (dnes již netradičně) napsán na psacím stroji. Písařce bude nějakou dobu trvat, než jej přeťuká do počítače. Grafik bude chvíli přemýšlet nad podobou knihy a vy pak strávíte nějaký čas převodem grafického návrhu do LaTeXových maker. Během té doby již může písařka přepisovat rukopis a používat standardní makra jako `\chapter` a pro kontrolní výtisk použít běžnou třídu BOOK. Mezitím vytvoříte jiný styl, kde bude makru `\chapter` přiřazena jiná vizuální reprezentace.

Kontextové značkování je velmi přínosné i pro redaktory časopisů a sborníků. Časopis má určitou koncepci a měl by být vizuálně jednotný. Pokud autoři důsledně používají kontextové značkování, spočívá sjednocování v předefinování poměrně malého množství maker. Je-li ovšem

článek zaplněn vizuálními značkami, nezbývá, než se v textu důkladně povrtat.

LaTeX je dosti silně založen právě na kontextovém značkování. Proto je paradoxně jednodušší přesazení typograficky ohavného LaTeXového článku s důsledně kontextovým značkováním než sjednocení dobře vysázeného článku v plain TeXu, který se hemží vizuálními makry `\vskip`, `\sevenrm` a jinými.

Vizuální značkování však má své použití. Sázíte-li tiráž nebo titulní stránku knihy, bylo by nesmyslné zdržovat se vývojem kontextových maker. Pokud ale hodláte sázet celou ediční řadu s jednotným vzhledem, vyplatí se vytvoření obecného stylu s definicemi kontextových maker pro titulní stranu, tiráž, vydavatelský záznam, jakož i další části knihy.

6. Změna vzhledu LaTeXových dokumentů

Oklikou jsme se dostali znovu tam, kde jsme před jedenácti stranami začínali. V celém seriálu nás bude zajímat vizuální vzhled dokumentu. U článků a knih budeme chtít obsah, ale graficky jinak vyvedený. Chceme tedy použít `\chapter`, `\section`, `\tableofcontents`, ale rádi bychom změnili vizuální reprezentaci. V dalších dílech si tedy řekneme, jakým způsobem toho lze snadno dosáhnout.

Autor má již poměrně rozsáhlý archiv definic použitých při sazbě různých knih. Další makra vznikají při tvorbě tohoto zpravodaje. To vše poslouží jako ilustrace možností LaTeXu. Výklad rozhodně nebude čistě akademický, ale bude založen na příkladech, které byly skutečně použity.

V příštím díle si povíme něco o možnosti použití jiných fontů než Computer Modern, o změně vzhledu obsahu a možná i o dalších problémech.

7. Doporučená literatura

První díl byl zároveň posledním, který vysvětloval primitivní záležitosti jako `\vspace` a `\@`. Autor očekává, že se čtenáři sami za domácí úkol naučí základy LaTeXu například z dokumentů, které jsou dodávány jako součást balíku CsTeX.

Pro vážné zájemce o LaTeX bych doporučil knihu: Michel Goossens, Frank Mittelbach, Alexander Samarin – *The LaTeX Companion.* Addison

Wesley, Reading 1994, ISBN 0-201-54199-8. Seriál „LaTeXová kuchařka“ ovšem vzniká nezávisle na této knize. Některé části mohou být duplikovány, avšak v seriálu se bude psát i o tom, co ve zmíněné knize nenajdete. Naproti tomu LaTeX Companion popisuje leccos, o čem se v seriálu psát nebude.

Uvedli jsme, že uživatel plain TeXu má plnou kontrolu nad veškerými mechanismy. To se projevuje také tím, že pomocí primitivu `\def` lze definovat složitá makra, na která `\newcommand` nestačí. Pokud chcete využívat LaTeX na 100 %, musíte si stejně prostudovat základní TeXovou bibli, tedy: Donald E. Knuth – *The TeXbook*. Addison Wesley, Reading 1984. ISBN 0-201-13448-9.

Zdeněk Wagner
`wagner@mbox.cesnet.cz`

Následující dva texty vznikly jako semestrální práce z předmětu „Publikační systém TeX“, který přednáší Petr Olšák na Elektrotechnické fakultě ČVUT. Po drobných, zejména jazykových, úpravách jsou zařazeny do našeho bulletinu, přičemž studenti (autoři jednotlivých příspěvků) s touto formou zveřejnění souhlasí.

Instalace TeXu na výkonově „malých“ PC

Jaroslav Řezníček

1. Motivace

„Jó, je to pohoda,“ řekl jsem si, když jsem usedl před klávesnici PC 486, na němž byl již naistalován emTeX. Člověk „to“ jen spustí a už může vesele a relativně rychle pracovat na sazbě větších dokumentů.

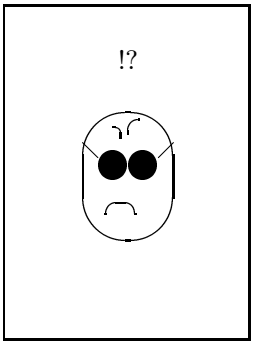
Představte si však situaci, že vlastníte kromě všelijakého jiného haraburdí i nějaké to „maličké PC“, jehož hardwarové vybavení je opravdu

chudičké. Všehovšudy je obdařeno procesorem 8086, popř. 8088, v nejlepším případě V30, dále LCD displejem s grafickým adaptérem CGA s maximálním rozlišením 640×200*, jedním megabytem operační paměti a asi tak dvaceti megabyty na pevném disku.*

U takovéhle mašinky, do které nemůžete „vecpat“ ani ty Windows, narazíte celkem brzy na problém, jak s její pomocí zpracovávat texty alespoň trochu solidně. Poznáte mnoho softwarových produktů, které vyžadují tu procesor alespoň 286 (to by u V30 ani tak moc nevadilo), tu grafický adaptér jiný než CGA, tu více operační paměti, tu více místa na disku, atp. Pomalu ale jistě vás začne pokoušet myšlenka, k čemu že vlastně je na té malé mašince paralelní port? Pokud na něj přeci jen připojíte třeba laserovou tiskárnu a nehledíte na to, že ty dvě věci vypadají vedle sebe jako David a Goliáš, pak se pro vás TEX stane příjemnou možností, kterak tu tiskárnu přimět k něčemu solidnímu.

Nedejte se proto zmást prvotními neúspěchy při svých pokusech o instalaci TEXu na takovéto „malé mašinky“ a přečtěte si alespoň další dvě kapitoly tohoto textu. Pokud máte více času, prohlédněte si i následující ukázky cvičení v TEXu!

1.1. Radost z fungujícího TEXu vyjadřujeme různě...



Vážení přátelé, tento obrázek je jedním z prvních, které jsem z radosti nad právě se rozběhnuvším TEXem nakreslil. Věnuji jej proto s úctou a láskou všem „dělníkům TEXu“...

Prostředí `picture` není zrovna to, v čem by chtěl člověk kreslit obrázky, ale uznejte, ve chvíli, kdy se vám jiného grafického software příliš nedostává, to k zahození určitě není.

1.2. Škola hrou

Je jistě užitečné vyzkoušet si alespoň něco málo z matematického aparátu TEXu. Navíc toto je semestrální práce z předmětu ohodnoceného matematickými kredity, takže přece jen...

$$\frac{(-1)^{1023}}{\log_2^{255}\frac{\left(\sqrt[17]{1^{1024}}+0^7\right)}{\text{počet hodin strávených nad TEXem}}} = \text{velmi malinkaté kladné číslo}$$

2. Ale teď už vážně!

Zdálo by se, že instalace TEXu na malá PC bude stejně snadná, jako je tomu u PC velkých, pouze mnohem pomalejší. Nenechte se ale tímto prvním dojmem mýlit! Jak poznáte z dalšího textu, nemusí to být vždy jednoduchá záležitost, obzvláště v případě, kdy se proti vám spojí všechny možné hardwarové nevýhody takového miniaturního PC.

2.1. Pokus o klasickou instalaci

Pokud není váš malý počítač vybaven žádnou floppy mechanikou a vy si nehodláte žádnou externí v nejbližší době pořídit, nastanou první komplikace. Klasická instalace TEXu by mohla probíhat snad tak, že byste si potřebné instalační soubory (postačily by diskety 1 až 4, viz dále) nejprve po kabelu nakopírovali na disk, a pak spustili instalaci. Jenže ta spočívá v tom, že se potřebné soubory nejprve zkopírují z instalačních adresářů do pracovních, a pak se teprve rozbalí. Jestliže nechcete zasahovat do instalačních dávek, nezbyde vám, než si kvůli takovéto instalaci vyhradit na disku dočasně téměř třikrát více místa, než bude zabírat výsledná instalace, což vyvolá nutnost nemilých zásahů do pěkně plného disku.

V případě, že se vám i toto podaří obětovat, spustíte samotnou instalaci a zažijete možná nepříjemnou zkušenost, kdy vám instalační dávka ihned po spuštění „zboří“ systém! Třeba je to tím, že máte na tomto počítači v ROM paměti DOS, který se navenek tváří jako MS-DOS 5.0, ale s jeho skutečnou kompatibilitou je to jaksi horší. Co se mne týče, změnil jsem po těchto neúspěších své plány a ustoupil od standardní instalace TEXu.

2.2. Přijatelná varianta instalace

Nejvhodnějším způsobem, jak nainstalovat TEX na malé PC výše zmíněného typu, je nainstalovat jej na velkém PC s ohledem na budoucí konfiguraci a výslednou strukturu přenést po kabelu na cílový disk. Ušetří se tak nemálo času i úsilí. Následující text je návodem, jak postupovat při takovéto instalaci.

Na disku velkého počítače vytvoříme adresář pro TEX se stejnou cestou, jaká bude na cílovém disku (použijeme např. příkaz `subst`), a spustíme klasickou instalaci (`install`) do tohoto adresáře. Z nabídek si postupně vybereme *malý TEX pro 8086*, zdrojové texty (např. *plain, LATEX*), fonty *80, 240 i 300 dpi* (kladně reagujeme pouze na první dotazy na zmíněné fonty, vzniknou tak prázdné knihovny), fonty samotné nechceme instalovat žádné. O tom, které budeme potřebovat, můžeme rozhodnout později a vygenerovat si je. Dále budeme chtít instalovat např. editor *Qedit*, *vlnkovací program*, *csindex*, *texcad* a program *mnu*. Pokud nechceme používat vlastní ovladač české klávesnice, můžeme nainstalovat také *kbd*. Jako výchozí verzi TEXu označíme `TeX` a METAFONTu `SBmf`. Nainstalujeme též *zdrojové texty fontů CSTEXu*.

Instalační program pak nakopíruje na disk soubory `.ZIP` z disket:

1. `nsetX`, `tfm`, `texinput`, `cstocs`, `cstugdoc`, `plain`, `latex`, `uvodlat`, `mnu`, `mfinput`, `fontlib`, `cfg`,
2. `qedit`, `tie`, `tex`, `latexdoc`, `scr`, `dot`,
3. `csindex`, `texcad`, `sbmf`,
4. `mfware`, `zdroje`.

Po rozbalovací fázi zbývá už jen překopírovat cca 9 MB instalovaného TEXu na cílový počítač a tam instalaci doladit.

2.3. Doladění instalace

Aktivace spouštěcích dávek Při dalším postupu je výhodné řídit se pokyny v souboru `CODELAT.TXT` (v adresáři `TEX\BATS`), tedy nejprve překopírujeme dávky `PLAIN.BAT`, `LATEX.BAT`, `INITEX.BAT` z adresáře `TEX\BATS` někam, kam ukazují cesty. Pak spustíme dávku `INITEX` a vygenerujeme `FMT` binární soubory pro všechny nainstalované formáty, v tomto případě tedy plain a LATEX, přičemž jako verzi TEXu uvedeme malý TEX.

Konfigurace sbmf Dle pokynů v dávce `BAS.BAT` (v adresáři `TEX\MFINPUT`) upravíme soubory:
`LOCAL.MF` – `localfont:=epsonfx`
`EGA.MF` – `screen_rows:=200;base_version:=base_version&"/CGA";`
pro rozlišení 240 dpi a grafický adaptér CGA. Po nastavení
`set METAFONT=sbmf`
spustíme z TEXmenu dávku `BAS.BAT`, která generuje báze *plain* a *cm* pro SBMF.

Nyní je nutno nakonfigurovat samotný SBMF tak, že spustíme:

```
sbmfset.exe sbmf.exe
```

a nastavíme:

```
default directory for sbmf bases: TEX\SBBASE\
default directory for sbmf inputs: TEX\MFINPUTS\
graphics mode for sbmf: 2: CGA 640x200
```

Generování fontů Generování potřebných fontů zajistíme nastavením parametrů v menu `Metafont`. Pokud potřebujeme vygenerovat např. font `cstt10` v rozlišení 240 dpi, pak vyplníme údaje takto:

```
%MF% file (w/o extension): cstt10
METAFONT &plain \mode=epsonfx; mag=1.0; input %MF%
Convertor gftopk %MF%.240 %MF%.pk
```

Následující tabulka ukazuje časy v sekundách, jak dlouho je potřeba na vygenerování jednotlivých fontů **v rozlišení 240 dpi**. Jsou tam uvedeny právě všechny fonty, které je nutno použít k tisku tohoto dokumentu.

Celkový čas v sec	Velikost písma v pt			
Zkratka pro písmo	5	7	10	12
cmmi	–	248	252	–
cmsy	–	104	108	–
csbx	–	–	439	437
cscsc	–	–	440	–
csr	414	436	445	–
csti	–	–	405	–
cstt	–	–	418	–
lcircle	–	–	34	–

Předchozí údaje nám chtějí naznačit, že generování jednoho fontu o jedné velikosti nám zabere na malém PC řádově minuty. S časy naměřenými na větších PC 486 je proto, prosím, raději nesrovnávejte...

3. Některé postřehy na závěr

3.1. Automatické dogenerování fontů

V souboru `CFG.DOC` (v adresáři `TEX\CFG`) je zmínka o tom, jak zajistit, aby prohlížeč v případě nenalezení potřebného fontu vyvolal jeho automatické dogenerování. Vytvořil jsem proto příslušnou dávku `TEXCFG.DAT` podle návodu, ale automatického generování fontů jsem nedocílil. Zůstává otázkou, jestli jsem kdesi cosi opomněl, nebo je to důsledek použití takovéhoto PC.

Na místě je otázka, zda je při takto pomalé konfiguraci vhodné používat automatické generování fontů. Pak je lepší volit při sazbě již dříve vygenerované a osvědčené fonty, přičemž nové generování spustíme ručně v případě, že font opravdu nutně potřebujeme nebo jej hodláme v budoucnu hojněji využívat. Máme tak jistě lepší kontrolu nad volnou kapacitou svého disku.

3.2. Nemají se rádi...

Toto tvrzení platí o „mém malém počítači“ a programech na prohlížení dvi souborů. (Chyba bude nejspíš v samotném DVISCR.) Nejenže se mi nepodařilo prohlížeč přimět k tomu, aby přešel na CGA režim 320×200 (prohlížení v režimu 640×200 je velmi nevěrohodné, protože trpí tzv. *somálským syndromem*), navíc při každém pokusu o standardní ukončení prohlížeče (i spuštěného z TEXmenu) tento chladnokrevně zničí můj grafický režim. Nepomáhá ani `cls`, ani přímé nastavení `textového módu`. Snad jediné, co pomáhá, je nejprve nastavit nějaký grafický režim, a potom požadovaný textový. To spolehlivě obstará následující prográmek v assembleru.

Pokud chceme s prohlížečem pracovat pohodlně, nezbyde nám, než tento pomocný prográmek spustit automaticky po každém ukončení prohlížeče, tedy např. změnit originální dávku `TEXSET.BAT` (v adresáři `TEX\CFG\RAM`) tak, že za každý řádek

```
%TEXDIR%\dvidrv dviscr @scr.cnf %SCROPT% %MAIN%
```

připíšeme řádek s voláním tohoto `.COM` programku:

```
CODE    SEGMENT BYTE PUBLIC
        ORG     100H
Start:
        MOV     AX,4
        INT     10H
        MOV     AX,3
        INT     10H
        MOV     AX,4C00H
        INT     21H

CODE    ENDS
        END     Start
```

Literatura

[L1] P. Olšák: *Typografický systém TEX*, CSTUG 1995.
[L2] A. Balvínová, M. Bílý: *Sázecí systém LATEX* (cvičení), ČVUT 1994.

Jaroslav Řezníček
xreznice@cslab.felk.cvut.cz
5. ročník

TEX na počítačích Apple Macintosh

Petr Šebek

Úvod

Na počítačích Apple Macintosh existuje několik verzí publikačního systému TEX: CMacTEX, OzTEX a DirectTEX. Zmíněné programy jsou buď freeware, nebo shareware. Mezi těmito programy jsou různé odlišnosti, které vyplývají z následujícího přehledu.

CMacTEX:

- docela rychlý
- hlavní prostředí obsahuje editor a překladač (2 MB RAM)
- DVI zobrazovač je samostatná aplikace (1 MB RAM)
- obsahuje různé pomocné programy a METAFONT
- obtížnější instalace
- CMAcTEX je freeware, ale „velká“ verze je shareware (25 USD)

OzTEX:

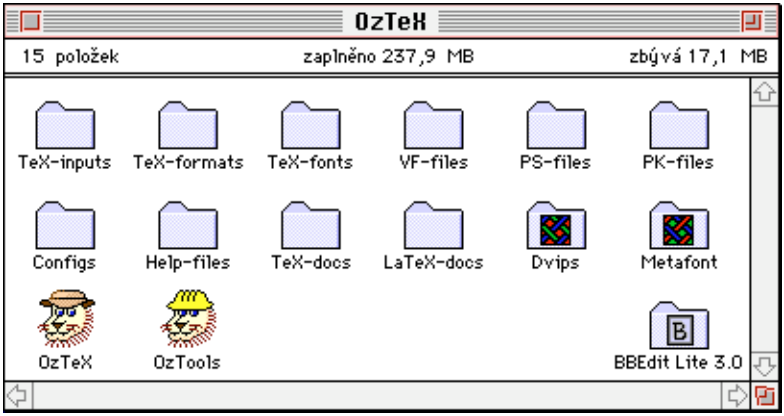
- poměrně rychlý
- hlavní prostředí obsahuje překladač a DVI zobrazovač (1,5 MB RAM)
- editor je volitelný
- obsahuje málo pomocných programů
- `dvips` a METAFONT vyžadují složitější instalaci (shareware 25 USD)
- OzTEX se instaluje snadno
- OzTEX je shareware (30 USD)

DirectTEX:

- velmi rychlý
- hlavní prostředí obsahuje editor, překladač a zobrazovač (1,5 MB RAM)
- obsahuje různé pomocné programy a METAFONT
- poměrně snadná instalace
- DirectTEX je shareware

Ze sítě Internet jsem získal na pěti disketách program OzTEX a na devíti disketách program CMacTEX. Po převedení souborů z internetovského formátu na formát Macintosh jsem rozbalil na disk všechny soubory, které byly na disketách. Nejprve jsem se pokusil nainstalovat CMacTEX, ale po asi hodině pokusů jsem toho zanechal. Potom jsem se pustil do instalace OzTEXu. Zde jsem nalezl jeden soubor, který se jmenoval „Correct“. V tomto souboru byl asi tento obrázek, který představuje příklad správné instalace.

Po uložení všech složek a souborů podle obrázku byl OzTEX nainstalován. Dále jsem pracoval pouze s programem OzTEX. Proto se další popis omezuje pouze OzTEX.



Obrázek 1: Takto vypadá správně nainstalovaný OzTEX

OzTEX

Po spuštění OzTEXu se objeví okno, které vidíte na obrázku 2.

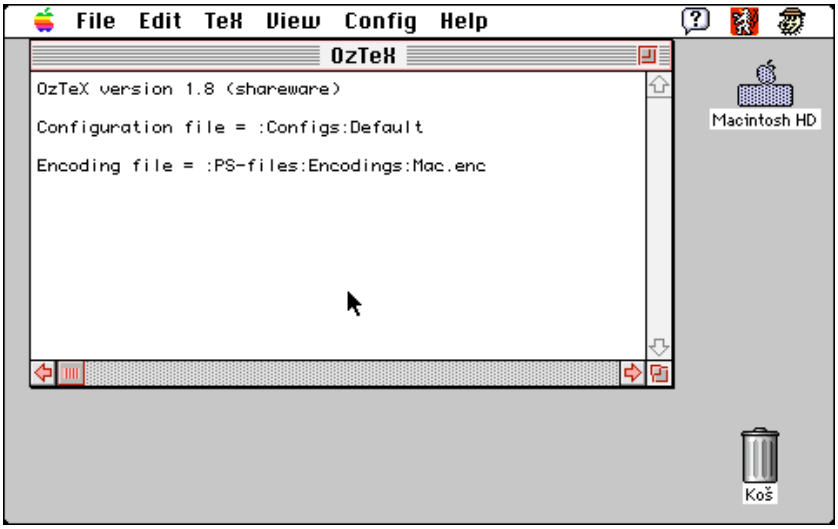
V okně s názvem OzTeX se objevují všechna hlášení a zprávy OzTEXu. Do tohoto okna nelze psát, ale text z něj lze kopírovat do schránky. Okno také nelze zavřít. Součástí OzTEXu není integrovaný editor zdrojových textů, ale spolu s OzTEXem je dodáván textový editor BBEdit Lite 3.0. V OzTEXu je možné zvolit libovolný externí editor, lze tedy například používat Microsoft Word nebo jiný textový editor. V prostředí OzTEXu však je integrován prohlížeč DVI souborů a nástroj pro jejich tisk na PostScriptových i ne-PostScriptových tiskárnách. Myslím, že možnosti OzTEXu pomůže přiblížit popis jednotlivých menu.

OzTEX menu

Nabídka File (Soubor)

Nabídka obsahuje následující položky:

Print DVI... vytiskne zvolený DVI soubor. Po zvolení této nabídky se otevře dialogové okno, ve kterém lze zvolit předem vytvořený DVI soubor, který bude vytištěn na připojené tiskárně.



Obrázek 2: Otisk obrazovky po spuštění OzTEXu

Obrázek 3: Nabídka File (Soubor)

Print ?.dvi vytiskne naposledy vytvořený DVI soubor (místo otazníku je uvedeno jméno naposledy vytvořeného DVI souboru).

Print Text... vytiskne zvolený textový soubor.

Page Setup... otevře standardní dialogové okno pro nastavení parametrů připojené tiskárny.

Use Standard PostScript – pokud je tato položka zatržena, je pro tisk použit standardní PostScript.

Use `dvips` – pokud je tato položka zatržena, zavolá se při tisku `dvips` pro překlad DVI souboru do PostScriptu.

Send PostScript pošle zvolený textový soubor do připojené tiskárny jako PostScriptový program.

Save OzTeX Window... uloží obsah okna OzTeX jako text do zvoleného souboru.

Quit ukončí OzTEX.

Obrázek 4: Nabídka Edit (Úpravy)

Nabídka Edit (Úpravy)

Nabídka obsahuje kromě standardních položek **Undo, Cut, Copy, Paste, Clear** také následující položky:

Copy OzTeX Window zkopíruje obsah okna OzTeX do schránky.
Font umožňuje změnit font v okně OzTeX.
Size umožňuje změnu velikosti fontu v okně OzTeX.
Choose Editor... umožňuje zvolit externí editor pro editaci zdrojových textů.
Select Error Line – pokud dojde k chybě a zvolený textový editor podporuje událost Otevřít dokument se zvláštními parametry, označí se v editoru číslo řádky, na které došlo k chybě.
Edit ?.tex otevře ve zvoleném editoru zdrojový text naposledy TEXovaného souboru.

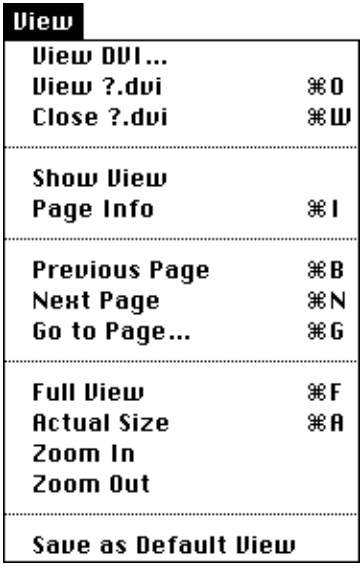
Obrázek 5: Nabídka TeX

Nabídka TeX

Nabídka obsahuje tyto položky:
TeX... spustí TEX na zvolený textový soubor. OzTEXové okno se bude chovat jako terminál.
TeX ?.tex spustí TEX na naposledy TEXovaný soubor (otazník je opět nahrazen jménem naposledy TEXovaného souboru).
Use Format in Log – pokud je tato položka zatržena, pokusí se OzTEX před spuštěním TEXu načíst formát uložený v příslušném `log` souboru (pokud tento soubor existuje).
Fix VFs in DVI... zamění ve zvoleném DVI souboru virtuální fonty za příslušné znaky a příkazy z aktuálních fontů.

Fix VFs after TeX – pokud je tato položka zatržena, připojí OzTEX automaticky všechny virtuální fonty, které byly vytvořeny v DVI souboru na konci běhu TEXu.

INITEX spustí INITEX.



Obrázek 6: Nabídka View (Zobraz)

Nabídka View (Zobraz)

Nabídka se skládá z těchto položek:

View DVI... zobrazí zvolený DVI soubor na stínítko monitoru. Po zvolení této položky se objeví dialogové okno, ve kterém lze volit číslo první zobrazené stránky, zvětšení a další parametry.

View ?.DVI zobrazí naposledy vytvořený DVI soubor na stínítko monitoru (místo otazníku je uvedeno jméno naposledy vytvořeného DVI souboru).

Close ?.DVI zavře okno, ve kterém je zobrazen naposledy vytvořený DVI soubor (místo otazníku je uvedeno jméno naposledy vytvořeného DVI souboru).

Show View – pokud není okno zobrazující DVI soubor na popředí, přesune se tam.

Page Info vypíše do okna OzTeX informace o právě zobrazené stránce jako je její velikost, velikost vložených obrázků, použité fonty a další informace.

Previous Page zobrazí předcházející stránku zobrazeného DVI souboru.

Next Page zobrazí následující stránku zobrazeného DVI souboru.

Go to Page... zobrazí stránku zadaného čísla zobrazeného DVI souboru.

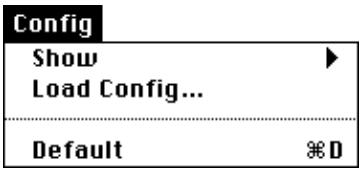
Full View zobrazí stránku zobrazeného DVI souboru tak, aby byla vidět celá.

Actual Size zobrazí stránku zobrazeného DVI souboru tak, aby se co nejvíce blížila skutečné velikosti.

Zoom In zvětší měřítko zobrazení aktuálního DVI souboru.

Zoom Out zmenší měřítko zobrazení aktuálního DVI souboru.

Save as Default View uloží měřítko zobrazení a další parametry jako default.



Obrázek 7: Nabídka Config (Konfigurace)

Nabídka Config (Konfigurace)

Nabídka obsahuje následující položky:

Show zobrazí v okně OzTEXu současné hodnoty parametrů. Tato položka má další podpoložky:

- All (všechny),
- Folders and Files (složky a soubory),

- TeX Parameters (TeX parametry),
- PostScript Fonts (PostScriptové fonty),
- Encodings (kódování).

Load Config... (Nahraj konfiguraci) nahraje zvolený konfigurační soubor.

Default nastaví standardní hodnoty konfiguračních parametrů. Dále jsou v této nabídce uvedena jména konfigurací pro různá výstupní zařízení, např. `US Letter, A5 Portrait, A4 Landscape, View at 144 dpi, Linotronic, StyleWriter, ImageWriter`.

OzTEX a čeština

Protože je tento dokument napsán v češtině, je to důkazem toho, že OzTEX češtinu „zvládá“. Pro zobrazování češtiny však používá 7-bitové kódování znaků. To znamená, že všechna diakritická znaménka jsou „dokreslována“ odděleně od písmen. Všechny znaky, které mají ASCII kód větší než 128, tedy zvláště česká písmena, mají předefinován `\catcode` jako aktivní znak. Nejprve se tedy vytiskne diakritické znaménko (háček, čárka, kroužek) a pod něj samotné písmeno. Tento přístup psaní češtiny však omezuje správnou funkčnost některých příkazů TEXu jako je třídění slov, změna na velká nebo malá písmena, rozdělování slov. Je to tedy vlastně řešení přechodné.

▷Příklad příkazu, který nebude proveden správně:

```
\uppercase{ješitná ježibaba}
```

jehož výsledkem je:

JEšITNá JEžIBABA

ale správně by měl být výsledek tento:

JEŠITNÁ JEŽIBABA

▷Příklad předefinování catcode českých znaků:

```
\catcode135=13 \def á{\'a}
\catcode137=13 \def Č{\v C}
\catcode139=13 \def č{\v c}
\catcode142=13 \def é{\'e}
\catcode145=13 \def Ď{\v D}
\catcode146=13 \def í{\'\i}
```

```
\catcode147=13 \def ď{\softd}
\catcode151=13 \def ó{\'o}
\catcode157=13 \def Ě{\v E}
\catcode158=13 \def ě{\v e}
\catcode190=13 \def Í{\'I}
\catcode197=13 \def Ň{\v N}
\catcode203=13 \def ň{\v n}
\catcode222=13 \def ř{\v r}
\catcode225=13 \def Š{\v S}
\catcode228=13 \def š{\v s}
\catcode231=13 \def Á{\'A}
\catcode232=13 \def Ť{\v T}
\catcode233=13 \def ť{\softt}
\catcode234=13 \def Í{\'I}
\catcode235=13 \def Ž{\v Z}
\catcode236=13 \def ž{\v z}
\catcode238=13 \def Ő{\' O}
\catcode241=13 \def Ů{\accent23U}
\catcode242=13 \def Ű{\'U}
\catcode243=13 \def ů{\accent23u}
\catcode248=13 \def Ý{\'Y}
\catcode249=13 \def ý{\'y}
```

V tomto příkladu jsou použita makra `\softt` a `\softd`, která píší správné háčky nad písmena t a d. Výpis těchto maker neuvádím.

Závěr

Tento dokument byl napsán pomocí programu OzTEX verze 1.8 na počítači Macintosh LCIII 8 MB RAM, pevný disk 250 MB. Instalovaný OzTEX zabírá na pevném disku 17 MB. Dokument byl vytištěn na laserové PostScriptové tiskárně Hewlett-Packard LaserJet 4 ML (300 dpi). Zdrojové texty dokumentu byly pořízeny v textovém editoru BBEdit Lite 3.0. Zpracování dokumentu trvalo programu OzTEX 11,5 s. Při práci s OzTEXem se nevyskytly žádné větší problémy, proto jsem s programem spokojen.

TUGboat 17 (1) March 1996

The Czech SGML User's Group

The Czech SGML User's Group has been initiated with the purpose of being a forum for spreading knowledge and information about the Standard Generalized Markup Language (SGML) and related standards, and to support development that leads to an active use of SGML.

Membership is voluntary and free, by becoming a member you receive the following benefits:

- Member access to the SGML Czech Republic Forum at the WWW.
- Member access to the SGML Czech Republic Coference to be held once a year.
- Join the news mailing list.
- Receive abstracts on recently introduced SGML literature.
- Member access to the Groups FTP server with SGML related tools.
- Member access to the Groups Knowledge database.
- See the work of other colleagues on SGML in this country.

The group is aimed to be a source of information to decision makers in both academic institutions and information system experts that work in diverse industries that due to its nature, require solid, updated systems for Logistics, Workflow, Database management and documentation.

The group organizes the annual SGML Czech Republic Forum. Additional encounters with members can be coordinated through the board.

It is desired that members prepare a paper on their particular activities related to SGML of minimum 300 words of size. To prepare and submit a presentation to the annual Forum is also welcomed.

Further information is available at

`http://www.mtranslations.cz/sgml.htm`

or through email:

`arturo@mtranslations.cz`

I hope it will be of your interest, feel free contact me if you have any questions.

Best regards
Arturo

TEX Live

I am very glad to announce the launch today in Paris of TEX Live, a new CD published by the TEX Users Group, the UK TEX Users Group and GUIenberg (French TEX Users), with help from NTG (Dutch TEX Users) and many individuals from other groups.

TeX Live's 649 megabytes contains:

- a ready to run Unix TEX setup, Thomas Esser's teTeX (based on Karl Berry's Web2c). It has binaries for:
  - Linux on Intel and m68k platforms;
  - Irix 5.2, 5.3 on MIPS (SGI Indy/Indigo)
  - SunOS 4.1.3 on Sun
  - Solaris 2.3, 2.4, 2.5 on SPARC
  - HPUX 9.01, 10.01 for HP workstations
  - Digital Unix (OSF/1) 2.0 and 3.2 for DEC Alpha machines
  - FreeBSD and NetBSD on Intel platforms
  - Ultrix 4.3, for DEC Decstation machines
  - AIX 3.2, 4.1.1, for IBM RS6000 machines
  - NeXTStep on Intel platforms
- a very large support tree of macros, fonts and documentation arranged according to the TEX Directory Structure layout
- the GUTenberg Mac, DOS and Windows distributions (archived)

You can use the TEX system by running directly from the CD, installing on your hard disk, or by adding packages to your existing system.

This is not a dump of CTAN full of compressed archives. This is a *working* system. To make it useable under Unix, it uses the Rock Ridge extensions to the ISO9660 file system. Ordinary systems can still read it, but will not see the long file names or symbolic links.

If you are on any flavour of Unix, BUY this CD! There is no complicated compilation or moving of installed files around, it will just *WORK*.

More details, and ordering information, can be found at `http://www.tug.org/texlive.html`. If you dont have WWW access, mail me or tug@tug.org for details. Cost is around $ 20 for members of any TEX users group, or $ 40 for others (the prices vary, depending on postage).

All profits from sales go back to fund new versions of the CD, and to TEX-related development projects.

Sebastian Rahtz
Secretary, TeX Users Group

☞ CSTUG nabízí svým členům hromadný nákup CD TEX Live za 700 Kč. Objednávky můžete zasílat písemně i elektronicky na adresu sdružení uvedenou v tiráži.

☞ Dále nabízíme videokazetu VHS s názvem „Donald E. Knuth in Brno“. Tato kazeta zachycuje ve čtyřech hodinách dvě přednášky o generování náhodných čísel a o Stanford Graphbase, dále promoci a promoční řeč D. E. Knutha. Videokazetu lze objednat na FI MU, Botanická 68a, 60200 Brno. Cena videokazety je 900 Kč.

☞ CSTUG ještě nabízí omezené množství CD ROMů „4allTEX“ za cenu 600 Kč. Lze je objednat na adrese sdružení (viz tiráž), slovenští zájemci je mohou objednat též na adrese: Nadácia Jura Hronca, MÚ SAV, Štefánikova 49, 81473 Bratislava.

Objednávky lze též zasílat elektronicky na adresu `orders@cstug.cz`. *Ve výše uvedených cenách není zahrnuto poštovné a balné.*

V příštím čísle Zpravodaje najdete „Často kladené otázky“. Je to překlad anglického dokumentu „Frequently Asked Questions“.

Vydalo:	Československé sdružení uživatelů TEXu vlastním nákladem jako interní publikaci
Obálka:	Bohumil Bednář
Počet výtisků:	650
Uzávěrka:	31. května 1996
Odpovědný redaktor:	Zdeněk Wagner
Tisk a distribuce:	KOPP, Šumavská 3, 370 01 České Budějovice
Adresa:	CSTUG, c/o FI MU, Botanická 68a, 602 00 Brno
fax:	05–412 125 68
e-mail:	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

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

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

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

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

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

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

Podávání novinových zásilek povoleno Oblastní správou pošt
v Českých Budějovicích, j. zn. P 3.202/94 ze dne 19. července 1994