Zpravodaj Československého sdružení uživatelů TeXu

Han The Thanh

Alternativní výstup programu TeX – PDF

Zpravodaj Československého sdružení uživatelů TeXu, Vol. 6 (1996), No. 2, 69–85

Persistent URL: http://dml.cz/dmlcz/149757

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