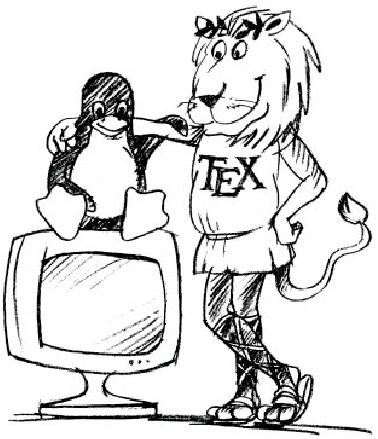
CSTUG



ZPRAVODAJ

Československého sdružení uživatelů TEXu

4

2000

ISSN 1211-6661

Ročník 10

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 přístupném archívu dostupném přes `http://www.cstug.cz`.

Své příspěvky do Zpravodaje můžete zasílat v elektronické podobě anonymním ftp na `ftp.icpf.cas.cz` do adresáře `/wagner/incoming/`, nejlépe jako jeden archivní soubor (`.zip, .arj, .tar.gz`). Současně zašlete elektronickou poštou upozornění na `mailto:bulletin@cstug.cz`. Uvedený adresář je přístupný pouze pro zápis. Pokud nemáte přístup na Internet, můžete zaslat příspěvek na disketě na adresu:

Zdeněk Wagner
Vinohradská 114
130 00 Praha 3

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

ISSN 1211-6661

Seminář o Linuxu a TEXu 2001

David Novák

Nebyli jste tam? Tak to jste udělali chybu. Takto by se dal jednou větou zhodnotit seminář o Linuxu a TEXu SLT 2001, který se konal v prostorách hotelu Skalský dvůr nedaleko Bystřice nad Pernštejnem na Českomoravské vrchovině. Seminář byl organizován sdruženími CZLUG a CSTUG a proběhl ve dnech 16.–18. února 2001. Informace o semináři je možno nalézt na `www.cstug.cz/slt/01`.

Den první

Kvůli špatnému spojení na místo konání přijížděli mnozí účastníci už ve čtvrtek během odpoledne a večera, ale oficiální zahájení SLT se konalo až ráno v pátek 16. února. Bezprostředně po krátkém uvítání účastníků organizátory začal dopolední blok přednášek. Příspěvky semináře se netýkaly pouze TEXových nebo Linuxových systémů, ale také nejrůznějších forem elektronického publikování, novinek ve světě internetu nebo Open Source programů obecně.

Úvodní příspěvek měla paní Jana Chlebíková z Univerzity Komenského v Bratislavě společně s panem Petrem Olšákem a týkal se nám dobře známého CD TEXLive5 a projektu TEXLive vůbec. I čtenáři Zpravodaje 1–3/2000 se dozvěděli nové informace, např. o problému se stále nedořešenou neslučitelností národních stylů `czech.sty` a `slovak.sty` se sadou maker Babel. Prezentace byla doplněna praktickými ukázkami fungování systému pod Linuxem.

Následovala velice zajímavá přednáška o generování vzorů dělení slov pro UNICODE (např. pro systém Ω) a o projektu `PATLIB` Davida Antoše a Petra Sojky z Fakulty informatiky MU v Brně, který by měl tento problém řešit. Přednášející David Antoš nás seznámil s některými netriviálními aspekty tohoto problému, s architekturou připravovaného systému a zmínil se o některých konkrétních implementačních záležitostech.

Redaktor Zpravodaje CSTUGu Zdeněk Wagner se s námi v dalším příspěvku podělil o své zkušenosti ze sazby tohoto občasníku. Zaujala mě hlavně část o lokálních definicích maker, o řešení problému více definic jednoho makra v balících použitých v dokumentu a o automatizaci zveřejňování obsahu Zpravodaje na WWW.

Další dvě přednášky se týkaly sazby matematiky. V první z nich mluvil bývalý předseda CSTUGu pan Karel Horák o rozdílech v některých typografických pravidlech českého (a většinou i evropského) standardu na jedné straně a amerického standardu AMS, který je (v podstatě) implementován v algoritmech TEXu

na straně druhé. Přednášející ukázal řešení některých problémů, které z toho plynou, a představil několik dalších sad znaků, které je možné využít pro sazbu nejen matematiky.

Ve druhé přednášce byl panem Michalem Marvanem ze Slezské univerzity v Opavě představen jím vyvinutý TEXový styl pro sazbu matematiky, který by měl odstranit vznikající propast mezi laicky používaným TEXem některými vědci a tradiční typografií dodržující jistá pravidla pro sazbu matematiky. Styl např. předefinovává LATEXovský příkaz `\frac` a primitivum `\over` tak, že styl sám rozhodne, zda se vysází $\frac{u}{v}$ nebo u/v a v případě nutnosti uzavře čitatel, jmenovatel nebo celý zlomek do závorek, aby se nezměnil matematický smysl zlomku.

Tím skončil dopolední blok přednášek a šlo se na oběd. Odpoledne bylo, pro změnu, věnováno spíše Linuxu a s ním spojenému software. Nejdříve nás pan Jakub Steiner seznámil s kouzly, která jdou dělat pomocí nové verze GIMPu, známého freewarového editoru rastrů v barevném modelu RGB. Jak jsme měli možnost poznat, jsou jeho možnosti minimálně srovnatelné s komerčními programy tohoto zaměření.

Následující příspěvek Reného Michálka se zabýval trendem komponentových technologií a konkrétně systémem HyperQbs. Zdá se, že právě komponentové programování je směr, kterým se vývoj softwaru bude v budoucnu ubírat. Psaní zdrojového textu už asi pomalu odzvonilo, teď budeme aplikace pohodlně sestavovat z připravených součástek.

Jakub Dadák z firmy Brain Systems s. r. o. měl přednášku s poměrně širokým záběrem s názvem „Linux ve světě e-business aplikací“. Hovořil o několika technologiích používaných pod Linuxem pro high-availability systémy na internetu, jako je HTTP server Apache, aplikační prostředí pro on-line elektronický obchod (konkrétně IBM WebSphere Application Server), atd.

Základním kamenem každého internetového serveru – dobře fungující databází – se zabýval Milan Šorm z Ústavu informatiky PEF MZLU v Brně. Představil v současné době asi nejuznávanější databázi RDBMS Oracle, která má jedinou nevýhodu – není to freeware. Podělil se s námi o cenné zkušenosti získané instalací a správou této databáze na MZLU a porovnal ji s jinými, volně šiřitelnými systémy.

Velice zajímavé komplexní řešení serveru prezentoval Michal Mühlpachr z firmy Nextra. Aplikační server Zope v sobě integruje všechny vrstvy používané u profesionálních web serverů – databázi (zde přístupnou přes URL), HTTP server, prostředí pro vývoj dynamických web aplikací, atd. Používání i správa Zope je velice pohodlná a přístupná odkudkoli přes WWW rozhraní; není nutný přímý přístup do souborového systému.

Po přestávce na večeři pokračoval program přednáškou, která vyvolala bouřlivou, asi hodinu a půl trvající, diskusi. Pan Stanislav Polčák přečetl příspěvek slečny Lucie Rambouskové (oba jsou z právní divize firmy Mironet) o novém

autorském zákonu, který v ČR platí od 1. 12. 2000. Jak se zdá, jsou v zákonu chyby, které by mohly mít fatální dopad na jistotu, že freeware je zadarmo. Hrozí totiž vznik kolektivního správce pro software, který by měl právo vybírat poplatky za jakékoli veřejné provozování jakéhokoli software, bez ohledu na názor majitele licence tohoto software. Vznikající veřejné sdružení autorů software *zastudena.cz* (`www.zastudena.cz`) vyvíjí snahu, aby takovýto kolektivní správce vůbec nevznikl, případně aby se jím stalo právě sdružení *zastudena.cz*.

Poslední páteční příspěvek měl Radomír Kurečka a týkal se problémové situace v oblasti certifikace znalostí Linuxu v ČR. Tato služba sice u nás existuje, není to nijak levná záležitost a školící střediska, poskytující kromě školení i přezkoušení, nemají žádnou společnou koncepci. Autor přednášky navrhl projekt vzniku českého certifikátu znalostí Linuxu, který by měl vzniknout s přispěním CZLUGu a být jím garantován. O tomto projektu se jednalo i v sobotu na valné hromadě CZLUGu.

Den druhý

Velká část sobotního dopoledne byla věnována poměrně novému, ale stále více populárnímu, značkovacímu jazyku XML (eXtensible Markup Language), vhodnému jednak pro elektronické publikování, ale také pro výměnu a sdílení dat a pro elektronický obchod. Přednášejícím byl jeden z nejerudovanějších českých odborníků na toto téma, pan Jiří Kosek. XML je vlastně metajazyk pro definici dalších jazyků, který vznikl zjednodušením SGML. Už z této definice plyne, že to je velmi otevřený a flexibilní systém. Uživatel XML si ukládá data do vlastní logické struktury, tvořené libovolnými tagy podobného tvaru jako v HTML. Definice vizuální reprezentace takovéto struktury je oddělena a rozhodně se v takovém souboru lépe vyhledávají informace, než třeba v TEXovém zdrojáku. Jednou z dalších výhod XML je interní práce s 32bitovými znaky (ISO 10646 Unicode).

Bezprostředně následující přednáška (stejného autora) se týkala formátování XML dokumentů. K tomu se používá speciální jazyk XSL (eXtensible Stylesheet Language). V tomto jazyce s velice intuitivní a průhlednou syntaxí se do samostatného stylového souboru definuje převod nějakého XML formátu např. do HTML, TEXu, nebo jakékoli jiné podoby. PassiveTEX je balík maker, který umožňuje sazbu XML dokumentů s kvalitou TEXového výstupu; jeho principy a použití pan Kosek také demonstroval. Tento systém zcela odstiňuje uživatele od makrojazyka TEXu, ale výsledek je perfektní.

Jan Pazdziora z FI MUNI v Brně se ve svém příspěvku také zabýval XML. V informačním systému Masarykovy univerzity se XML rozhraní používá ke striktnímu oddělení generování dat od jejich formátování. Informace získané z databáze jsou skriptem převedeny do XML struktury předem definovaného tvaru. Teprve potom jsou pomocí XSLT mechanismů zkonvertovány do HTML

podle separátního formátovacího souboru (takže např. každý uživatel systému může mít vlastní design stránek). Tento systém je pak poměrně jednoduše rozšiřitelný např. pro mobilní WAP rozhraní nebo jiné výstupy.

IP verze 6 bylo téma přednášky populárního autora knih o IT Pavla Satrapy. Čtyři bajty adresy IPv4 už zkrátka pomalu přestávají stačit; IPv6 těch bajtů má 16, prý by to mělo stačit už „navždy“. Příspěvek se zabýval strukturou adresy a datagramů, směrováním, bezpečnostními mechanismy, přímou podporou mobilních zařízení, také aktuálním stavem implementací IPv6 ve stávajícím IPv4-prostředí a mechanismem bezbolestného přechodu z IPv4 na IPv6.

Sobotní odpoledne bylo koncipováno velice volně. Lidé si vytvořili různé sekce a v těch „něco dělali“. Jedna z největších sekcí byla sekce turistická, jejíž činnost spočívala v asi tříhodinové procházce po rozbahněných a zledovatělých cestách (většinou lesem) a v dokonalém vymrznutí zapříčiněném studeným větrem. Zkrátka několik zástupců druhu *homo computerus* na výletě.

Ke konci odpoledne měl pan Michal Bulant přednášku o kryptografii. O tento příspěvek jsem bohužel přišel, ale vím, že se zabýval historií kryptografie, jejími principy a některými ze současných využití, jako je elektronický podpis nebo volby.

Po večeři až do pozdních nočních hodin probíhala valná hromada sdružení CZLUG a už od odpoledne také zasedal výbor CSTUGu.

Den třetí

Neděle byla posledním dnem semináře a přednášky probíhaly jen dopoledne. První z nich se týkala adresářových služeb standardu LDAP a tuto poměrně novou technologii pro správu např. konfigurací uživatelů, jejich práv nebo skupin uživatelů nám představil Michal Mühlpachr. Tento systém sdružuje evidenci a správu pro větší množství aplikací (web, mail, pop servery i správu uživatelů na úrovni OS), takže není potřeba udržovat speciální konfigurační soubory/formáty pro každou aplikaci zvlášť.

Pan Tomáš Hála z MZLU se ve svém příspěvku zabýval bibliografickými citacemi. Popsal normu, týkající se tvaru bibliografických citací a ukázal ji na příkladech. Také se zmínil o aktuální otázce citování elektronických zdrojů.

Že je Linux vhodný také jako součást rozsáhlejších systému nám ve své přednášce nazvané *Clusterová řešení na Linuxu* dokázal Jan „Yenya“ Kasprzak z FI MUNI v Brně. *Clusterem* v tomto kontextu rozumíme skupinu víceméně samostatných počítačů spojených do jednoho funkčního celku. Přednášející nejdříve popsal základní typy clusterů, používaný software a nakonec se s námi podělil o konkrétní zkušenosti s clustery.

V předposledním příspěvku představil Michal Kvasnička z Ekonomicko-správní fakulty MU v Brně velice mocný TEXový formát ConTEXt. Tento systém

vyvíjený panem Hansem Hagenem z Holandska je určen především pro mechanickou sazbu velkých elektronických dokumentů. Má přímou podporu PDF včetně možnosti využití JavaScriptu a také propojení s METAPOSTem – generování obrázků za chodu.

Martin Mareš je členem týmu vyvíjejícího systém BIRD Internet Routing Daemon a přišel o něm pohovořit. Idea tohoto projektu je vytvoření směrovacího daemona, který by podporoval několik routovacích protokolů, IPv4 i IPv6, umožňoval dynamickou rekonfiguraci, udržoval si více routovacích tabulek a měl ještě spoustu dalších krásných vlastností. Zdá se, že se jim to docela daří, tak jim budeme držet palce.

Tak už mi věříte, že jste udělali chybu, pokud jste na SLT 2001 nebyli? Kromě skvělého nabitého programu semináře, mohu pět chválu i na prostředí, kde jsme ty tři dny strávili. Skalský dvůr je velice hezký hotel, který je dostatečně vzdálen od ruchu civilizace. Na jídlo si určitě také nikdo stěžovat nemohl. Je potřeba vzdát díky všem organizátorům v čele s předsedy obou pořádajících sdružení Janem Kasprzakem a Petrem Sojkou. Doufám, že se uvidíme na některém z dalších ročníků...

ConTEXt

Michal Kvasnička

Téměř žádný TEXista nepoužívá TEX v jeho „přirozené“ podobě, jak ji představují programy `initex` a `virtex`. Místo toho používá určitý *formát* – balík maker, který někdo připravil, aby mu zjednodušil práci. V České republice se dnes používají především dva takové formáty: plainTEX a LATEX. V blízké budoucnosti by se k nim mohl připojit ještě jeden: ConTEXt. A nejen připojit, ale možná i vytlačit LATEX z jeho dominantního postavení. Aby se tak mohlo stát, chci vás v tomto příspěvku s ConTEXtem rámcově seznámit.

ConTEXt vyvíjí soukromá nizozemská společnost Pragma ADE, jmenovitě pan Hans Hagen. Ačkoli vzniká na půdě komerční firmy, je celý ConTEXt (jeho TEXová, METAPOSTová a perlovská část) distribuována zdarma pod GNU licencí. Jediná omezení se týkají modifikace zdrojových souborů – ty se už po modifikaci nesmějí jmenovat stejně. Také dokumentace je šířena zdarma, a to ve dvou jazycích, holandsky a anglicky. Celý ConTEXt, včetně důležitých perlovských programů a dokumentace najdete na WWW stránkách společnosti Pragma ADE: `www.pragma-ade.com`. Rozhodně se vyplatí stahovat ho právě odtud, protože tak máte zajištěno, že i při velmi rychlém vývoji ConTEXtu získáte vždy poslední stabilní verzi.

ConTeXt je formát na podobně vysoké úrovni jako LaTeX v tom smyslu, že definuje určité logické struktury (např. kapitoly, oddíly, plovoucí prostředí pro obrázky apod.), se kterými pak dále pracuje (např. vytváří obsahy, seznamy obrázků apod.). Proto budu v tomto textu srovnávat ConTeXt právě s LaTeXem.

Od LaTeXu se ConTeXt liší především svým zaměřením. Zatímco LaTeX je určen především pro sazbu vědeckých článků, kde dokonalý vzhled dokumentu není primárně důležitý (naopak je spíše vhodné zachovávat určitý standardní design), ConTeXt je určen především pro sazbu rozsáhlých elektronických dokumentů, ve kterých design hraje významnou roli. Odtud vyplývá i hlavní rozdíl mezi LaTeXem a ConTeXtem: na rozdíl od LaTeXu, kde téměř každá změna vzhledu jednotlivých logických elementů vyžaduje „kutání“ hluboko pod povrchem (každý, kdo se skutečně snažil změnit vzhled výčtového prostředí, o tom ví své), jsou změny designu v ConTeXtu velmi snadné. Téměř ke každému makru (v terminologii ConTeXtu příkaz, command), které ovlivňuje vzhled dokumentu, existuje speciální příkaz, který upraví vzhled příslušného logického elementu.

Dalším výrazným rozdílem oproti LaTeXu je fakt, že ConTeXt používá externí balíky jen zcela výjimečně. Všechny důležité vlastnosti jsou integrovány do „jádra“. Jádro je pak na TeXové zvyklosti poměrně velké (formát současné anglické verze ConTeXtu přeložený pod Linuxem pro pdfeTeX má téměř přesně 3,5 MB; formát LaTeXu má za stejných okolností asi 660 KB). Výhodou tohoto přístupu je to, že odpadá klasický problém se sdílením externích balíků: pokud někomu pošlu zdrojový text svého dokumentu, stačí přiložit pouze *mé vlastní* soubory a říct, jaká nejstarší verze jádra je potřebná ke kompilaci.

Od plainTeXu a LaTeXu se ConTeXt liší ještě jedním rysem: obvykle se nespouští přímo binární TeX (program `virtex`), ale speciální perlovský program `texexec`. Ten se spolu s pomocným programem `texutil` stará o několik věcí: aby byl dokument zkompilován v potřebném počtu průběhů, aby byly zkompilovány a vloženy pomocné soubory s rejstříky, obsahy, METAPOSTovými triky apod. Zároveň se také stará o správu projektů, módů a verzí (viz dále).

Dalšími silnými stránkami ConTeXtu je také vynikající podpora PDF (a to včetně vkládání JavaScriptu) a neuvěřitelně silná spolupráce s METAPOSTem. Než se však dostaneme k těmto specialitám, podívejme se nejdříve blíže na některé „klasické“ vlastnosti ConTeXtu.

Design dokumentu jako celku

Jak jsem už řekl, změnit vzhled dokumentu je v ConTeXtu nesmírně jednoduché. K tomuto účelu slouží celá škála pomocných příkazů, jejichž jména vesměs začínají (v anglické verzi) slovem `\setup`. Obvykle se jmenují `\setup`*něco*, pokud nastavují vlastnosti jednoho příkazu, nebo `\setup`*něco*`s`, pokud mění cho-

vání celé třídy příkazů. Parametry se uvádějí do hranatých závorek (na rozdíl od LaTeXu hranaté závorky *neznamenají* nepovinné parametry, ale *netextové* parametry). Jednotlivé parametry se obvykle oddělují čárkami. Pokud nějaký parametr může mít hodnotu, přidělí se mu pomocí rovnítka. Dále v textu uvedu několik příkladů.

Podívejme se nejdříve na způsob, jak ovlivnit vzhled dokumentu jako celku. ConTeXt umožňuje zadat na začátku práce velikost papíru, a to jak velikost papíru, na který se sází, tak velikost papíru, na který bude tisková strana umístěna. Např. příkaz

```
\setuppapersize[A4][A4]
```

zajistí, že se sází na papír o velikosti A4 a výsledek je umístěn na papír téže velikosti. Naproti tomu příkaz

```
\setuppapersize[A5][A4]
```

sází stránky o velikosti A5 a umístí je na papír o velikosti A4. To se hodí např. pro korekční tisky, kdy je třeba vyznačit ořezání stránek (jak se přidají ořezové značky, ukážu později). Samozřejmě je možné definovat novou velikost papíru, pokud žádná z předdefinovaných velikostí nevyhovuje sazečovým potřebám.

Příkaz `\setuppapersize` však dokáže ještě mnohem víc: dokáže stránky otáčet o 90, 180 a 270 stupňů, sázet podélně (landscape) nebo převést celý tisk do negativu, případně ho *ozrcadlit* (mirror, tj. převrátit tisk okolo svislé osy).

Ve spolupráci s příkazem `\setuparranging` lze stránky na úrovni TeXu dokonce *přeskládat a sloučit*! To např. umožňuje vytvářet sborníky o velikosti A5 i v případě, že výstupem je PDF, a není tedy možné použít programy z balíku psutils. Příkaz dokáže jak změnit pořadí stránek, tak umístit několik sazebních stran na jednu výstupní.

Vlastní kompozici stránky popisuje příkaz `\setuplayout`. Stránka je rozdělena do několika částí. Shora dolů je to vršek (top), záhlaví (header), textové tělo, zápatí (footer) a spodek stránky. Mezi záhlavím (zápatím) a textem může být také dodatečná vzdálenost. Zleva doprava je to (pro pravé stránky) hřbetní mezera (backspace), v ní umístěné dva různé levé okraje (left margin a left edge), textové tělo a dva různé pravé okraje. Příkaz `\setuplayout` nastavuje velikosti všech těchto částí stránky naráz. A nejen to: dokáže také nastavit umístění „vnitřní stránky“ na papíře (například tehdy, když sázím na A5 a tisknu na A4), zapnout tisk ořezových značek nebo zapnout sazbu na řádkový rejstřík. Ano, čtete dobře: pokud je třeba sázet na řádkový rejstřík, stačí v ConTeXtu prostě zapnout jeden přepínač. Všechny (nebo téměř všechny) příkazy se tomu automaticky přizpůsobí; to málo, co zbývá, jde s pomocí příslušných příkazů ošetřit ručně.

Místo detailního popisu se podívejme na jednoduchý příklad. Předpokládejme, že chceme sázet do textového sloupce o šířce 13 cm a 41 řádcích na

stránku, bez záhlaví. Textový sloupec je na papíře umístěn 4 cm shora a 5 cm zleva. Zápatí má výšku dvou řádků a navazuje těsně na textový sloupec. Navíc ještě sázíme na řádkový rejstřík. Tohoto designu dosáhneme snadno nastavením

```
\setuplayout
  [topspace=4cm, backspace=5cm,
   width=13cm, lines=41,
   header=0pt, footer=2\baselineskip,
   grid=yes]
```

Pokud bychom navíc chtěli zapnout tisk ořezových značek, stačí do nastavení přidat parametr `marking=on`. Nastavení `marking=color` přidá navíc za hranice ořezu barevnou a černobílou škálu.

Nastavení jednotlivých částí stránky můžeme vizuálně zkontrolovat pomocí příkazu `\showframe`. Jestli ConTeXt opravdu dodržuje řádkový rejstřík, se můžeme přesvědčit pomocí makra `\showgrid`. Po jejich zadání jsou na výstupu vidět boxy ohraničující jednotlivé oblasti stránky, respektive účaří jednotlivých řádků.

Rozdělení stránky do jednotlivých oblastí však neovlivňuje pouze vlastní sazbu. V ConTeXtu je možné nastavit také *pozadí* každé části stránky. Pozadí může tvořit téměř cokoli: text, barevná výplň, obrázek, interaktivní menu nebo nějaká jejich kombinace. Podobným způsobem lze snadno umístit na každou stránku např. logo společnosti nebo projektu.

Pokud bych např. chtěl, aby byl pravý okraj na pravých stránkách (a levý okraj na levých) orámovaný a vybarvený žlutě, mohl bych nastavit

```
\setupbackgrounds[text][rightmargin]
  [frame=on,
  background=color,
  backgroundcolor=yellow]
```

Pokud bych na místo toho chtěl, aby byl na pozadí celé stránky nějaký obrázek, mohu (při výstupu do PDF) zadat

```
\defineoverlay[pozadi]
  [{\externalfigure[desert2.jpg]
  [width=\overlaywidth, height=\overlayheight]}]
\setupbackgrounds[page][background=pozadi]
```

První příkaz vytvoří tzv. *overlay*, tj. „překrývající plochu“, která bude v tomto případě obsahovat obrázek. (Obecně může overlay obsahovat téměř cokoli.) Velikost obrázku se automaticky přizpůsobí velikosti overlaye. Druhým příkazem se overlay vloží na pozadí stránky. V tuto chvíli se spočítá jeho skutečná velikost a obrázek se podle ní přizpůsobí. Mechanismus vkládání overlayů a jiných typů

pozadí se neomezuje pouze na části stránky – ve skutečnosti valná většina vizuálních elementů umožňuje tento mechanismus standardním způsobem využít. To se týká např. i poznámek pod čarou nebo poznámek na okraj. Na pozadí každého z těchto elementů je možné vložit libovolný počet overlayů.

Při výstupu do PDF je možné příkazy `\setuppapersize` a `\setuplayout` poněkud zneužít a použít znovu na každé nové stránce. Tak lze dosáhnout zvláštního efektu, kdy je každá strana jinak velká a jinak orientovaná. Pokud nebude změněno nastavení pozadí, automaticky se samo přizpůsobí zadaným změnám.

Podobným způsobem je pomocí dalších `\setup` příkazů možné nastavit i další komponenty designu, jako je obsah záhlaví, patičky a způsob číslování stran. Jiné příkazy nastavují rozteč tiskových řádků, velikost odstavcové zarážky a její umístění (např. zda má být odstavec po vynechaném řádku odsazen), toleranci sazby apod.

Fonty

Způsob práce s fonty je v ConTeXtu v určitém směru asi na půl cesty mezi plainTeXem a LaTeXem, některé vlastnosti však v LaTeXu nemají obdobu. Makra ConTeXtu zadefinují pro zvolenou rodinu písem a kódování příkazy, které použijí příslušný řez ve správné velikosti. K tomu slouží příkaz `\setupbodyfont`. Pokud např. použijeme na začátku dokumentu příkaz

```
\setupbodyfont[pos, 17pt]
```

načte ConTeXt standardní postscriptová písma (Times, Helveticu a Courier) a nastaví základní velikost písma na 17 pt. Na rozdíl od LaTeXu není uživatel ConTeXtu omezen na velikosti 10, 11 a 12 pt; zdá se, že jediným omezením jsou tu schopnosti TeXu a METAFONTu, což umožňuje velmi snadno vytvářet jak různé prezentace s velkými písmeny, tak slovníky s drobnými písmeny. Zároveň s nastavením standardní velikosti písma ConTeXt nastaví také vzdálenosti účaří standardních řádků a některé další údaje a modifikuje přepínače fontů. Od této chvíle `\tf` znamená „textfont“ (základní patkové písmo v základní velikosti), v našem případě Times Roman o velikosti 17 pt, `\it` znamená italiku tohoto písma, `\sl` skloněnou variantu tohoto písma, `\bf` tučnou atd. Příkaz `\bs` např. přepne sazbu do tučného skloněného písma.

ConTeXt nicméně neobsahuje plný ekvivalent NFSS, známého z LaTeXu. Pokud tedy chceme větší řez běžného textového písma, musíme využít příkazy jako `\tfa`, `\tfb` apod., pro menší řezy jsou k dispozici příkazy `\tfx` a `\tfxx`. Podobné příkazy existují i pro další textové varianty. Přepínání sazby mezi patkové, bezpatkové a neproporcionální písmo obstarávají příkazy `\rm`, `\ss` a `\tt` respektive.

Kromě standardních přepínačů existují také „rychlé“ přepínače, které přepnou sazbu přímo do požadovaného řezu. Mají obvykle poměrně intuitivní tvar, např. `\ssbf` přepne sazbu do tučného bezpatkového písma standardní velikosti.

Mimo to existuje i makro pro zdůrazněný text. Označuje se `\em`. Toto makro se automaticky přizpůsobuje použitému řezu. Navíc doplňuje automaticky i kurzívní korekci. Jeho použití je prosté:

```
{\em Toto zdůrazni.}
```

Standardně se v ConTeXtu ke zdůrazňování používá skloněné písmo. Pokud to chcete změnit na italiku, stačí zapsat příkaz

```
\setupbodyfontenvironment[default][em=italic]
```

K použití písem v ConTeXtu je třeba uvést ještě jednu poznámku. Na rozdíl od LaTeXu se ConTeXt snaží standardizovat i použití fontů. Pokud tedy chceme místo standardních anglických postscriptových fontů používat jejich české obdoby, neměli bychom vytvářet nový definiční soubor fontů (nějaké cs-pos), nýbrž sáhnout do speciálního lokalizačního souboru, který je součástí distribuce, a v něm vytvořit *synonyma* použitých fontů. Víc k této problematice je možné nalézt v dokumentaci a archivu konference o ConTeXtu.

Kapitoly, oddíly

Patrně nejdůležitějším standardním elementem téměř každého dokumentu jsou různé hlavičky: nadpisy kapitol, oddílů a pod. ConTeXt umožňuje nejen jednoduše měnit vzhled stávajících hlaviček, ale vytvářet i další. Tak může např. existovat několik různých hlaviček stejné úrovně (odpovídajících např. klasickému příkazu `\section`) odlišených vzhledem, zápisem do obsahu apod.

Vzhled každé hlavičky je možné měnit příkazem `\setuphead`. Protože tento příkaz má mnoho parametrů, ukážu raději dva jednoduché příklady. Nastavení

```
\setuphead[section]
  [style=bold,
   number=no,
   align=left,
   before={\blank[3*line, force]},
   after={\blank[2*line]},
   indentnext=yes]
```

zajistí, že oddíly (používá se příkaz `\section{...}`) budou mít následující vzhled: nad nadpisem se vynechají tři běžné řádky, pod ním dva. Nadpis nebude číslován. Bude vysázen tučným řezem běžného patkového písma, a to vpravo

(nenechte se zmýlit nastavením „left“, to v ConTeXtu obvykle znamená „doprava“ :–). První odstavec pod nadpisem bude mít odstavcovou zarážku (pokud jsou zapnuté).

Naproti tomu nastavení

```
\setuphead[chapter]
  [numberstyle={\bfa},
   textstyle={\bfd},
   page=right,
   header=empty,
   before={\blank[5*line,force]},
   after={\blank[3*line]},
   command=\mychapter]
\def\mychapter#1#2{%
  \vbox{%
  #1
  \blank[2*line]
  #2}}
```

způsobí, že kapitola bude umístěna na nejbližší nové pravé stránce. Číslo bude vysázené větším tučným, vlastní nadpis velkým tučným písmem. Na stránce bude potlačeno záhlaví. Před nadpisem bude vynecháno pět řádků, pod ním tři. Vlastní vzhled kapitoly určuje makro `\mychapter`.

Příkaz `\blank` se stará o vynechání místa. Jako parametr může mít buď přímo délkový údaj nebo celočíselný násobek výšky standardního řádku apod. Parametr `force` zajistí, že se mezera na začátku stránky neztratí.

Pokud sázíme na řádkový rejstřík, budou obě hlavičky automaticky upraveny tak, aby řádkový rejstřík nenarušily. Budou posazeny účařím posledního řádku na první možné účaří řádkového rejstříku.

Křížové odkazy

Druhou podobně důležitou skupinu příkazů tvoří příkazy, které se starají o křížové odkazy. Také pro ně platí, že jejich vzhled a chování lze velmi jednoduše modifikovat.

První část příkazů má na starost tvorbu obsahu, seznamu obrázků, tabulek a jiných plovoucích prostředí a rejstříků. V ConTeXtu je velmi snadné definovat další plovoucí prostředí a definovat jeho vlastní seznam. Stejně tak je možné definovat speciální rejstříky.

Příkaz, který slouží k modifikaci vzhledu obsahu a dalších seznamů, má tolik parametrů, že se raději zmíním jen o několika jeho užitečných vlastnostech. Obsahy mohou být lokální i globální. Lokální obsah zahrnuje jen seznam oddílů,

pododdílů atd. příslušné kapitoly (nebo oddílu ap.), globální obsah zahrnuje seznam všech hlaviček v celém dokumentu. V jednom dokumentu může být vytvořen jak globální obsah, tak lokální obsahy různých úrovní. Samozřejmě je možné určit, které typy hlaviček budou zařazeny do obsahu a které ne. Tento výčet se může na různých místech dokumentu lišit. Jedná se skutečně o *výčet*, takže zařazení do obsahu nemusí být nutně „spojité“ jako v LaTeXu. Pokud k tomu máme nějaký důvod, můžeme do obsahu zařadit třeba kapitolu a pododdíl, ale nikoli oddíl.

Vzhled obsahu lze snadno modifikovat: buď vybrat jednu z přednastavených variant, nebo nastavit chování každého prvku obsahu zvlášť. Stejná makra (`\setuplist`, resp. `\setupcombinedlist`) mohou při výstupu do PDF nastavit také interaktivitu položek obsahu, tj. označit, která část položky (číslo hlavičky, její popis, číslo stránky nebo celá položka) je interaktivní.

Podobně snadno modifikovatelné jsou i poznámky pod čarou (footnotes). Příslušný nastavovací příkaz dokáže snad všechno, co v LaTeXu dělají speciální balíky, včetně číslování poznámek pod čarou na každé straně zvlášť. Jedna z voleb dokonce umožňuje změnit poznámky pod čarou v poznámky na konci textu (endnotes). Ty mohou být také lokální, tj. rozdělené po kapitolách nebo jiných oddílech, nebo globální.

Klasické křížové odkazy jsou řešeny poněkud jinak, než je obvyklé v LaTeXu. Každý element, na který je možné se odkázat, má logické jméno odkazu jako jeden ze svých parametrů. Tak je možné napsat např.

```
\chapter[odkaz, sem]{Název kapitoly}
```

Nyní jsou definovány dva různé odkazy (`odkaz` a `sem`), které se vztahují k téže kapitole. V textu se pak na tuto kapitolu můžeme odkázat trojím způsobem, pomocí příkazů `\in`, `\at` a `\about`. Příkaz `\in` vypíše číslo kapitoly, příkaz `\at` vypíše číslo strany, na které se kapitola nachází, a příkaz `\about` vypíše název kapitoly. Všechny tři příkazy mají také nepovinné parametry, které mohou obsahovat text. Pokud je zapnutá interaktivita, slouží jako „cíl pro kliknutí“ nejen vlastní číslo, ale i tento text. Příkazy pak mohou být zapsány např. následujícím způsobem:

```
v~\in{kapitole}[sem]
```

Jako „cíl kliknutí“ pak slouží celý text „kapitole 5“. Obdobně fungují i odkazy na obrázky, tabulky apod. Samozřejmě je také možné vytvořit autonomní odkaz přímo na zvolenou stránku.

Mechanismus odkazů je ovšem ještě podstatně obecnější. Nejobecnějším příkazem je zde příkaz `\goto`. Ten může (při výstupu do PDF) obsahovat nejen výše zmíněné klasické křížové odkazy, ale také odkazy na další PDF dokumenty, některé speciální příkazy pro PDF prohlížeč a dokonce příkazy JavaScriptu.

To umožňuje velmi snadno vytvářet interaktivní dokumenty. Např. lze vytvořit „klikátko“, které načte WWW stránku, jiný PDF dokument, spustí hudbu nebo video, „roluje“ několik obrázků umístěných přes sebe, skočí na další nebo předchozí stránku, spustí v prohlížeči vyhledávání nebo ukončí prohlížeč. Jediné, co je třeba znát, je jméno speciálního odkazu.

Projekty, prostředí a módy

Každý LaTeXista ví, že LaTeXový dokument začíná příkazem `\documentclass`, který definuje, jaká třída dokumentu (předdefinovaný design) se má použít. Za tímto příkazem následuje nepovinná preambule, ve které se načítají další styly, které mají změnit design dokumentu. Vlastní obsah dokumentu je uzavřen mezi dva příkazy `\begin{document}` a `\end{document}`.

Naproti tomu ConTeXt žádný takový úvodní příkaz nezná. Načítání „standardních“ stylů nemá žádný smysl, protože design dokumentu lze snadno upravit každému dokumentu „na míru“.

Vlastní dokument je uzavřen v páru příkazů `\starttext` a `\stoptext`. Tato dvě makra však rozhodně *neodpovídají* LaTeXové dvojici `\begin{document}` a `\end{document}`. Příkazy `\starttext` a `\stoptext` totiž mohou být vzájemně *zanořené*. V tom případě se chovají v podstatě jako začátek a konec bloku.

Tato odlišnost je nesmírně důležitá. Umožňuje totiž vytvářet něco, čemu ConTeXt říká *projekty*. Jeden a tentýž dokument může být vysázen buď samostatně nebo jako součást většího celku. To je v ConTeXtu možné bez jakéhokoli zásahu do zdrojového textu. Jeden dokument se označí jako „projekt“. Ten pak bude obsahovat všechny ostatní jako své části. Jednotlivé dílčí dokumenty jsou označeny jako „produkty“ (product) nebo „komponenty“ (component). Projekt obsahuje pouze popis designu a odkazy na produkty, z nichž se skládá; stejně tak každý produkt může obsahovat odkazy na jednu nebo více komponent (komponenta může obsahovat odkazy na další dílčí komponenty). Přitom každý dílčí dokument obsahuje odkaz na řídící projekt. Když se překládá jen část celku, např. nějaký „produkt“, načtou se všechny definiční části příslušného projektu. Tak je možné využít design celku i v každé dílčí části.

K čemu tato funkce slouží? Například k tvorbě rozsáhlých počítačových manuálů, které mají popsat balík několika kooperujících programů. Popis jednoho programu je v naší terminologii produkt. Ten se skládá z jednotlivých kapitol, komponent. Všechny popisy dohromady tvoří projekt. Předpokládejme, že distribuujeme dokumentaci jak v tištěné podobě, tak na přiloženém CD ROMu. Tištěná příručka by měla v jednom svazku popsat celý balík programů. Zkompilujeme tedy soubor popisující projekt. Na CD ROM chceme naproti tomu uložit dokumentaci rozdělenou do několika souborů (abychom příliš nezatěžovali paměť počítače) – co program, to soubor. V tomto případě zkompilujeme

každý produkt zvlášť. Pokud se v manuálu vyskytnou závažnější chyby, můžeme přesázet příslušnou kapitolu – komponentu a vystavit ji jako errata na Internet.

Projekty nejsou jedinou možností jak modifikovat hotový dokument bez nutnosti zásahu do jeho zdrojového kódu. Další možností jsou módy (mode). Stále častěji je třeba jeden dokument vysázet několika různými způsoby: jiný vzhled musí mít tisk pro jazykovou korekturu, jiný preprint pro výslednou korekturu, ještě jiný tisk pro výstup na osvitovou jednotku. Stále častěji je také nutné vytvořit interaktivní verzi pro prohlížení na obrazovce. V LaTeXu je možné tyto varianty realizovat celkem snadno: buď zasáhneme do preambule zdrojového textu a změníme stylový balík, který se má použít, nebo vytvoříme několik různých „hlavních“ souborů, které pak načítají ostatní části dokumentu.

ConTeXtové řešení je jiné. Uživatel nespouští přímo TeX, nýbrž perlovský skript `texexec`. Ten dokáže také řídit, jaká verze dokumentu se má vytvořit. Základní dva přepínače určují, zda se vytvoří barevný nebo černobílý výstup (ConTeXt dokáže přepínat mezi barevnou a černobílou verzí) a zda se jedná o výstup do DVI nebo do PDF. Zvláštní přepínač dokáže také spustit zvolený *mód* sazby. V popisu designu může uživatel definovat různé módy: každý mód obsahuje příkazy, které se provedou speciálně v případě, že `texexec` tento mód zavolá. Popis designu je možné uložit do zvláštního souboru – tomu se v ConTeXtové terminologii říká *prostředí*.

Barvy a obrázky

TeX jako takový neumí pracovat s barvami. Umožňuje však vložit do dokumentu informace, které určitý postprocesor interpretuje jako barvy. Totéž se týká obrázků. Potíž je v tom, že dva dnes nejobvyklejší postprocesory, `dvips` pro výstup do PostScriptu a pdfTeX pro výstup do PDF jsou vzájemně poměrně nekompatibilní. Záleží pak na TeXovém formátu, jak se podaří práci s barvami a obrázky standardizovat. ConTeXt v této oblasti značně pokročil, ale i tak má jeho řešení k dokonalosti ještě daleko.

Co se týče barev, jejich použití je stejné při výstupu do PDF i do PostScriptu. ConTeXt umožňuje definovat barvy jak ve formátu RGB, tak CMYK, a těmito barvami sázet jak text, tak pozadí téměř všech objektů. Kromě jednotlivých barev umožňuje vytvářet i celé palety – soustavy dobře ladících barev s daným stupněm šedi. Několik palet je předdefinovaných. Navíc ConTeXt umožňuje zapínat a vypínat použití barev. Pokud je použití barev vypnuto, je text sázen černou barvou a barva pozadí je bílá. Také obrázky v METAPOSTu jsou převedeny do černobílé varianty. To umožňuje snadno vytvořit dvě verze dokumentu: černobílou pro tisk a barevnou pro prohlížení na obrazovce.

Standardně jsou barvy vypnuté. Zapnout je lze příkazem

```
\setupcolors[state=start]
```

Kromě varianty `start` existuje ještě několik dalších možností. Na tomto místě odkazuji na dokumentaci.

Práci s obrázky dominuje práce s METAPOSTovými obrázky. ConTeXt je dokáže vložit nejen do výstupu pro `dvips`, ale i do PDF. Potřebné konverze přitom proběhnou na úrovni TeXu. Tato makra přebírá i LaTeX. ConTeXt navíc dokáže konvertovat barevné METAPOSTové obrázky na černobílé. Bohužel to dělá přímo na úrovni výstupních METAPOSTových souborů, takže před kompilací barevného dokumentu je potřeba všechny METAPOSTové obrázky přegenerovat.

Vkládání ostatních obrázků je standardizováno aspoň natolik, že je možné psát jejich jména bez koncovek. ConTeXt si pak vybere tu variantu, která je pro daný výstup nejvhodnější. Při výstupu do PostScriptu bude preferovat obrázky s koncovkou `.eps`, při výstupu do PDF `.pdf` apod. Pokud neexistují, hledá další varianty obrázků v pořadí, které preferuje.

Samozřejmostí je „recyklace“ obrázků. Příkaz

```
\useexternalfigure[figgold][goldprice][width=5cm]
```

načte rozměry nejlepší varianty obrázku uloženého v souboru `goldprice.*` a zvětší nebo zmenší obrázek tak, aby byl široký 5 cm. Výsledek uloží do paměti pod logickým jménem *figgold*. Nyní můžeme tento obrázek vložit podle jeho logického jména příkazem

```
\externalfigure[figgold]
```

a případně opět zvětšit nebo zmenšit. Oba příkazy toho umějí ještě mnohem více. Mimo jiné dokážou vytvářet logická jména obrázků z jiných logických jmen. V tom případě se uplatní *dědičnost*: potomek převezme nastavení svého předka a liší se pouze těmi atributy, které jsou výslovně uvedeny. (Tato vlastnost se zdaleka netýká jen obrázků, ale také hlaviček oddílů a mnoha dalších elementů.) Pokud používáme výstup, který umí recyklovat obrázky (např. PDF), pak bude do výsledného dokumentu obrázek vložen fyzicky pouze jednou.

Některé další důležité vlastnosti

Než se podíváme na některé velmi speciální vlastnosti ConTeXtu, řekněme ještě několik slov o některých klasických „prostředích“, jmenovitě o sazbě do více sloupců, výčtových prostředích a tabulkách.

Z maker na sazbu do více sloupců se mi zdá být ConTeXtová verze nejstabilnější. Nedochází ani ke ztrátám textu jako v eplainu, ani k „přetékání“ stránek jako v LaTeXu. Po zapnutí sazby na řádkový rejstřík nejen souhlasí účaří ve všech sloupcích, ale řádkový rejstřík není narušen ani při přechodu mezi jedno a vícesloupcovou sazbou.

Plovoucí objekty jsou ve sloupcové sazbě buď vysázeny tam, kde jsou uvedeny, nebo, pokud to není možné, odplavou na následující stránku. Tam jsou umístěny přes tolik sloupců, kolik vyžaduje jejich skutečná šířka. Zdá se, že při tom nemohou být dva plovoucí obrázky na jedné stránce. Za určitých okolností může ConTeXt dokonce *přehodit* pořadí obrázků, pokud tato změna zlepší výsledný vzhled dokumentu.

Dalším důležitým mechanismem jsou *výčtová prostředí*. Standardní LaTeXová prostředí `itemize` a `enumerate` nahrazuje v ConTeXtu jediné prostředí jménem `\startitemize...\stopitemize`. Toto prostředí je nesmírně variabilní. Kromě něj existuje ještě několik dalších výčtových prostředí, která vesměs nemají v LaTeXu žádnou obdobu. Dokonce existuje i prostředí, které umožňuje sázet různé texty (např. různé jazykové verze) vedle sebe do několika sloupců tak, aby odpovídající si části textu byly vždy vedle sebe.

Dalším zajímavým mechanismem, který také nemá v LaTeXu obdobu, jsou *bloky*. Bloky umožňují na jednom místě napsat určitou část textu a vysázet ji na jiném, popř. vícekrát. To se hodí např. při sazbě učebnic. Otázky a odpovědi jsou napsány pohromadě. V příslušné kapitole se však tisknou pouze otázky, zatímco odpovědi (případně otázky i odpovědi) se tisknou až ve speciální příloze na konci knihy.

ConTeXt také umí automaticky zvýraznit (barevně nebo černobíle) syntaxi vybraných programovacích jazyků. Pokud vím, implementováno je barevné zvýraznění TeXu, METAFONTu, METAPOSTu, JavaScriptu, Perlu, SQL a nově také XML. Zdá se také, že není vůbec složité doprogramovat jednoduché barevné zvýraznění syntaxe *kteréhokoli* programovacího jazyka.

Další důležitou součástí většiny vědeckých a technických dokumentů jsou tabulky. ConTeXt má poměrně rozsáhlou podporu tabulek: existuje v něm několik různých prostředí, z nichž každé se vypořádává se sazbou tabulek jiným způsobem. Uživatel si může vybrat formu, která mu nejvíce vyhovuje. Mimo jiné zde existuje i prostředí, které umožňuje sázet tabulky způsobem, který je obvyklý v HTML, včetně slučování několika buněk na řádku nebo ve sloupci. Velkou silou tabulek je možnost formátování a barvení obsahu buněk. Na druhou stranu, možnosti rámování jsou mnohem omezenější než ve specializovaných balících LaTeXu.

Celkově lze říci, že každému prvku nebo prostředí známému z LaTeXu odpovídá v ConTeXtu jeden nebo více prvků a prostředí. Navíc v ConTeXtu existuje celá řada prostředí, která nemají v LaTeXu žádnou obdobu. Vzhled všech prvků lze relativně snadno modifikovat.

METAPOST a triky s grafikou

Jednou z nejsilnějších stránek ConTeXtu je jeho spolupráce s METAPOSTem. ConTeXt je s METAPOSTem tak provázaný, že Hans Hagen považoval za dobré napsat na toto téma velmi rozsáhlý manuál.

V čem tedy spočívá jejich spolupráce? Jednoduše vzato, ConTeXt dokáže za chodu uložit zvolenou část dokumentu do pracovního souboru spolu s METAPOSTovými formátovacími příkazy. Tento pomocný soubor je pak pomocí METAPOSTu zpracován a výsledek se opět načte v dalším průběhu TeXu. Uživatel se o pomocné soubory vůbec nemusí starat, protože jejich správu a kompilaci zajišťuje pomocný perlovský skript `texexec`.

K čemu je to celé dobré, vždyť obrázky se dají vytvářet v METAPOSTu přímo? Využití je celá řada. ConTeXt tímto způsobem řeší větší část problémů, které LaTeX ošetřuje pomocí balíků `graphics` a `graphicx`, např. zvětšování, zmenšování a rotace textu, ořezávání obrázků apod. Navíc se tento mechanismus uplatní všude tam, kde je třeba vzít část dokumentu a nějak ho graficky zpracovat. Např. můžeme chtít, aby na každé stránce byl černý obdélník se jménem kapitoly otočený oproti textu o 270 stupňů. Velikost obdélníku se musí změnit podle délky jména kapitoly. Nebo je třeba umístit nadpis kapitoly do elipsy, jejíž velikost se změní podle velikosti nadpisu (je zřejmé, že se elipsa musí nakreslit ve správné velikosti, a nikoli dodatečně zvětšovat, protože to by změnilo sílu čar). Nebo je třeba část textu, obrázků, tabulek atd. ořezat podle nějaké křivky. Tento mechanismus je také možné využít k orámování bloku textu nepravoúhlým rámečkem, rámečkem s popiskou a k mnoha dalším podobným věcem. Dále je možné vytvářet velké množství speciálních efektů potřebných pro elektronické publikace: např. ke tvorbě různých tlačítek, která se přizpůsobí velikosti svého obsahu, sazbu věty podél zvolené křivky, sazbu odstavce do tvaru popsaného křivkou apod. Také overlaye, které jsem zmínil dříve, mohou obsahovat METAPOSTové makro. Už to samo o sobě umožňuje neuvěřitelné efekty.

Větší část těchto problémů je teoreticky řešitelná i v LaTeXu – s pomocí balíku PSTricks nebo s pomocí METAPOSTu. Znamenalo by to ovšem spoustu programování (PSTricks navíc není možné použít při přímém výstupu do PDF). ConTeXt naproti tomu zajišťuje inteligentní rozhraní, které designerovi umožňuje soustředit se přímo na vlastní problém. Navíc je ke ConTeXtu přibalen i MetaFun – balík METAPOSTových maker, která dále zjednodušují tuto část sazby.

Další možnosti propojení TeXu a METAPOSTu představují moduly ppchTeX a Flowcharts. Ty umožňují snadno do TeXu integrovat chemické strukturní vzorce a strukturní diagramy, používané např. pro popis algoritmů.

Domnívám se, že toto šťastné propojení TeXu a METAPOSTu rozšiřuje schopnosti TeXu natolik, že se při zpracování grafických prvků nejen vyrovná, ale v mnoha směrech i předčí komerční „obrázkové“ WYSIWYG programy. Škála

grafických triků je totiž omezena pouze schopností sazeče programovat v METAPOSTu.

Interaktivní dokumenty a JavaScript

Díky panu Hàn Thế Thành ovi je dnes možné přimět TEX, aby generoval přímo nejen DVI, ale i PDF dokumenty. Od klasických formátů, jako je DVI nebo PostScript, se PDF v jednom směru dost podstatně liší: nejen že popisuje vzhled vlastní stránky, ale umožňuje do dokumentu integrovat i další speciální vlastnosti: PDF může obsahovat hypertextové odkazy, animace, přehrávat zvuky a dokonce spouštět i JavaScript. ConTEXt vytváří k těmto možnostem PDF dokumentů inteligentní rozhraní, takže je možné je relativně snadno začlenit do vlastního elektronického dokumentu.

Jak vytvořit hypertextový odkaz jsem popsal výše. Stačí zapnout interaktivitu, a všechny křížové odkazy se automaticky změní na hypertextové odkazy. Samozřejmé je, že lze nastavovat jejich vzhled (barvu, font atd.). Podobně snadno je možné vytvořit i speciální „tlačítka“ pro přechod na předchozí nebo následující stranu, pro skok na začátek nebo konec dokumentu, zavření dokumentu nebo ukončení prohlížeče, spuštění vyhledávání apod. K tomu všemu je možné využít buď výše popsaný příkaz `\goto` a speciální názvy křížových odkazů, nebo mechanismus *menu*.

Pomocí podobného mechanismu je možné propojit více PDF dokumentů dohromady tak, aby jeden spouštěl druhý. Také lze přimět dokument, aby načetl (pomocí externího prohlížeče) WWW stránku nebo poslal e-mail.

Také tvorba záložek (bookmarks) je snadná. Začlenění hlavičky kapitoly do záložek vyžaduje pouze zapnutí tohoto mechanismu a případně modifikaci výčtu těch hlaviček, které mají být do záložek zapisovány. Samozřejmě je možný i ruční zápis. ConTEXt sice podporuje automatickou konverzi nadpisů do Unicode (takže je zachována čeština), bohužel to ale zatím linuxový Acrobat Reader (verze 4.0) neumí, takže je lepší se tomu vyhnout – místo textu by uživatelé Linuxu viděli v záložkách pouze samé tečky.

Se začleňováním zvuků a videa nemám žádné zkušenosti. Ostatně, linuxový Acrobat Reader 4.0 zatím, zdá se mi, tyto vlastnosti nepodporuje.

Velice silnou stránkou je spolupráce ConTEXtu a JavaScriptu. ConTEXt jednak využívá JavaScript pro některé své vlastní cíle, jednak umožňuje uživateli začlenit do dokumentu kus vlastního kódu. ConTEXt používá JavaScript např. k rotování obrázků. Tato vlastnost může být velice výhodná např. při tvorbě prezentací s ekonomickou tématikou. V ekonomii je časté, že se určitá situace demonstruje sadou grafů, ve kterých se postupně různě posouvají křivky nabídky a poptávky. Rotace obrázků umožňuje to, že jednotlivé obrázky nejsou umístěné vedle sebe nebo na následujících stránkách, ale že se překrývají. Jednoduchý

přepínač (odkaz nebo tlačítko) pak zajistí buď zobrazení příslušné vrstvy, nebo jejich postupné zobrazování ve vhodném pořadí.

Kromě toho umožňuje ConTEXt vytvářet nejrůznější „políčka“ (fields): přepínací tlačítka (check buttons, radio buttons), vyplňovací políčka pro vstup textu apod. Obsah těchto políček může být spojen s nějakou proměnnou JavaScriptu a modifikovat jeho výpočet. Tlačítka, vytvořená pomocí příkazu `\goto` nebo pomocí mechanismu menu, mohou spouštět také vložené funkce JavaScriptu. Hans Hagen tímto způsobem naprogramoval funkční vědeckou kalkulačku (v PDF).

Zdá se, že tento mechanismus by mohl být vhodný pro vytváření různých interaktivních výukových programů, interaktivních testů, ceníků, které by samy počítaly ceny objednaných produktů (a případně i odeslaly e-mail s objednávkou) apod. Složitější aplikace ovšem vyžadují jednak znalost JavaScriptu, jednak spouštění na poměrně rychlých počítačích. PDF totiž pokaždé překresluje celou stránku znova. (Použití stránkové cache způsobuje v Linuxu problémy se zobrazením.)

XML

SGML a XML jsou v současné době skutečnou módní záležitostí. Zdá se, že jsou schopné vyřešit požadavek standardizace elektronických dokumentů bez nutnosti vnutit autorům jeden typ textového editoru (v prostředí českých univerzit by to byl nejspíše MS Word).

Protože jak SGML, tak XML jsou textově orientované strukturované popisy dokumentu, není principiálně velký problém vysázet je v TEXu. Hans Hagen publikoval článek [14], ve kterém demonstroval schopnost ConTEXtu sázet XML. Vysázel interaktivní přehled článků publikovaných v MAPSech (bulletinu nizozemského sdružení uživatelů TEXu) – vstupem mu byla bibliografická databáze kódovaná v XML. K tomu využil experimentální makra, která nejsou součástí běžné distribuce. Navíc bylo třeba XML dokument nejdříve pomocí speciálního parseru převést do podoby TEXových příkazů. Nicméně i tak byly výsledky nesmírně zajímavé. Jeho makra byla schopná obsloužit nejrozmanitější případy využití XML.

Zhruba v půli ledna 2001 však byla uvolněna nová beta-verze ConTEXtu, která obsahuje podporu *přímé* sazby XML v TEXu. Příkazy, které jsou k dispozici, popisuje [10].

ConTEXt nyní dokáže jak vysázet XML dokument uložený ve zvláštním souboru, tak přecházet mezi sazbou XML a TEXu v rámci jednoho dokumentu. Sazbu externího souboru s XML dokumentem umožňuje mimo několika jiných příkazů i příkaz `\processXMLfile`. Jeho použití je jednoduché

```
\processXMLfile{soubor.xml}
```

Přechod na sazbu XML umožňuje příkaz `\enableXML`. Zpětný přechod zajišťuje příkaz `\disableXML` – ten je však nutné zapsat ve formě XML značky, protože kategorie zpětného lomítka při sazbě XML už *není* 13. Lze tedy napsat např.

```
\enableXML
tady je kus <b>XML</b> dokumentu
<?context-command \disableXML ?>
```

Jednotlivým XML značkám je samozřejmě nutné přiřadit nějaký význam. (Značky, kterým nebyl žádný význam přidělen, jsou ignorovány.) ConTeXt definuje deset různých příkazů, které umožňují přidělovat XML značkám význam. Bere při tom ohled i na atributy. Další příkazy umožňují definovat význam entit.

Podívejme se na jednoduchý příklad, který rozhodně *nevyčerpá* možnosti ConTeXtu. Předpokládejme, že máme vysázet jednoduchý XML dokument, uložený v souboru `text.xml`, který vypadá takto:

```
<document>
  <title name="small">Primitivní dokument</title>

  <list packed="yes">
    <item>
      of course these commands <B>don't</b> match &context;
      commmand names
    </item>
    <item>
      and even worse, <ref name="att"> attributes will not
      be the same as in &context;
    </item>
  </list>
</document>
```

TeXovou interpretaci jednotlivých značek můžeme v ConTeXtu nastavit např. takto

```
\defineXMLenvironment [document] \starttext \stoptext
\defineXMLargument [title] \title
\defineXMLenvironment [list]
  {\startitemize[\XMLifequalelse{list}{packed}{yes}{packed}{}]}
  {\stopitemize}
\defineXMLenvironment [item] \item \par
\defineXMLenvironment [b] {\bgroup\bf} {\egroup}
\defineXMLentity [context] \ConTeXt
```

Význam je poměrně zřejmý. Příkaz `\defineXMLenvironment` definuje *prostředí*. Počáteční značku `<document>` TEX zamění za ConTEXtový příkaz `\starttext`, koncovou značku za `\stoptext`.

Naproti tomu příkaz `\defineXMLargument` „shrábne“ obsah značek a předá ho jako argument standardnímu ConTEXtovému příkazu `\title`. V našem případě bude výsledek stejný, jako bychom přímo v TEXovském dokumentu napsali příkaz `\title{Primitivní dokument}`.

Příkaz `\defineXMLentity` nastaví, že XML entita `&context;` se nahradí TEXovým příkazem `\ConTeXt`.

Zajímavé je také nastavení značky `<list>`. Ta se expanduje za standardní prostředí `\startitemize`. XML argument je však zpracován a předán příkazu. O argument se stará příkaz `\XMLifequalelse`. Zjistí, zda je zadaný argument (`packed`) značky (`list`) stejný jako zadaná hodnota (`yes`). Pokud tomu tak je, expanduje se čtvrtý parametr (v našem případě `packed`); v opačném případě pátý argument (tady prázdný). Tímto způsobem je možné převést XML nastavení parametru (`packed=yes`) do tvaru předpokládaného ConTEXtem (pouze `packed`).

Pokud bychom chtěli získat pouze hodnotu parametru, můžeme použít příkaz `\XMLpar`. Jeho použití je následující:

```
\XMLpar{title}{name}{}
```

Příkaz vrací hodnotu atributu `name` značky `title`. Třetí parametr může obsahovat implicitní hodnotu.

Soudě podle mých zkušeností (asi týden „hraní“ s podporou přímé sazby XML) jsou makra už v beta verzi poměrně stabilní. Problémy nastávají pouze v případě, kdy některé TEXovské makro spouštěné přes XML rozhraní nahrává vnější TEXovský soubor. Kategorie znaků jsou totiž změněné (např. zpětné lomítko už neznamená začátek kontrolní sekvence), a tak někdy dochází k podivným chybám. V mém případě k tomu došlo při nahrávání JavaScriptových preambulí a při vkládání METAPOSTových obrázků. Řešení je ovšem velmi jednoduché: kritická makra musejí na začátku nastavit správné (TEXem předpokládané) kategorie znaků a na konci opět přepnout kategorie znaků do sazby XML. V současné době Hans Hagen opravuje oba výše zmíněné případy, aby ke kolizím nedocházelo. Je tedy možné, že v první následující stabilní verzi už bude všechno v pořádku.

Lokalizace

ConTEXt je vcelku dobře připraven pro lokalizaci pro různé jazykové skupiny. V současné době existuje holandská, anglická, německá, italská a česká verze a připravují se další. Lokalizace v ConTEXtu znamená nejen překlad všech

standardních nadpisů (jako je Obsah, Seznam obrázků apod.) – ty už jsou do češtiny přeloženy, ale také překlad jmen *příkazů*. V české verzi (interface) se tedy místo příkazu `\chapter` používá název `\kapitola`. Naštěstí je možné tuto (podle mého mínění) nežádoucí vlastnost vypnout. Nejjednodušší (i když ne moc čisté) je před vygenerováním formátu nastavit v souboru `cont-cz.tex` definici `\def\defaultinterface{english}`.

Kromě toho je třeba počeštit také některé části pomocných programů. Perlovské programy `texexec` a `texutil` se totiž starají nejen o kompilaci dokumentu, ale také o vytváření křížových odkazů a rejstříku. Časem by se měly starat i o seznam literatury a nahradit tak program `bibtex`. Počeštění tvorby rejstříku ještě není hotovo. Podle dokumentace by mělo být snadno řešitelné na úrovni TEXu, ale zatím všechny moje pokusy selhaly. Buď se klíčová slova netřídila podle českého řazení písmen, nebo selhalo třídění velkých a malých písmen. Nicméně doufám, že i tato potíž bude v blízké době odstraněna.

Co ConTEXtu dosud chybí

ConTEXt toho ve své současné podobě umí opravdu hodně. Některé věci však ještě nejsou vyřešeny k plné spokojenosti. ConTEXt ještě nemá úplnou podporu `bibtex`u ani funkční obdobu $\mathcal{AMS}$-TEXu. Oba nedostatky prozatím řeší dva externí moduly, které ještě nejsou zcela dokonalé. Dalším problémem je tvorba českých rejstříků.

Jistou slabinou je také dokumentace. Některé kapitoly nebyly dosud z holandštiny přeloženy do angličtiny. Jindy manuál nepokrývá všechny vlastnosti popisovaných příkazů. To je však nutnou daní rychlého rozvoje a neustálého rozšiřování možností ConTEXtu.

ConTEXt také neumožňuje sázet dva akcenty nad sebe. Nicméně, jediné použití tohoto rysu, které mne napadá, je jméno pana Hàn Thế Thành. Pro jeho sazbu má však ConTEXt vlastní akronym: `\THANH`.

Závěr: srovnání plainu, LaTEXu a ConTEXtu

Zdá se mi užitečné porovnat plainTEX, LaTEX a ConTEXt z několika pohledů: podle rychlosti kompilace, snadnosti programování, „standardnosti“, přívětivosti k uživateli a grafické síly. Jako doplněk se může hodit i pohled na grafické preprocesory, postprocesory a jiné programy, které pomáhají při práci s TEXovým dokumentem. Podívejme se nyní na výsledky těchto tří formátů podle těchto kritérií.

Co se týče rychlosti kompilace, je plainTEX bezkonkurenčně nejrychlejší. LaTEX je o něco pomalejší a ConTEXt výrazně nejpomalejší ze všech. Taco

Hoekwater [11] srovnal rychlost sazby primitivního dokumentu (hladkého textu) o délce 200 stran v plainTEXu, LATEXu a ConTEXtu. Kompilace trvala 21 s, 27 s a 2 minuty a 59 s respektive. Podstatně pomalejší kompilace v ConTEXtu je daní především za mnohem komplexnější (a tedy i pomalejší) výstupní rutinu, která navíc v primitivní hladké sazbě zůstane vcelku nevyužita. Jistou daň si také vyžádá vysoká „parametrizovanost“ všech příkazů.

Co se týče „standardnosti“, standardem je dnes de facto LATEX. Pokud některé vědecké časopisy a konference přijímají příspěvky v TEXu, myslí se tím obvykle LATEX. Některé programy, především matematické jako je Maple, které dokážou generovat TEXovský výstup, také generují LATEX. Další silnou zbraní LATEXu je v této konkurenci LYX, vizuální preprocesor, jehož síla neustále roste. Nesmírně užitečný je také poměrně silný a stabilní konvertor z LATEXu do HTML. Ani plainTEX ani ConTEXt nemohou nic takového nabídnout.

Co se týče snadnosti programování a rozšiřitelnosti formátu, je na prvním místě opět plainTEX. Člověk, který má věci nejraději pod kontrolou, asi nebude používat složité makrobalíky, které se o všechno starají samy. Za ním bude patrně ConTEXt, který uživateli umožňuje, aby modulárně zařadil vlastní makra standardním způsobem do standardních příkazů. Navíc Hans Hagen téměř na požádání dodefinovává do ConTEXtu další užitečné funkce. LATEX z tohoto hlediska vychází, aspoň podle mého mínění, jako nejhůře přizpůsobitelný produkt.

Grafické a designerské možnosti má v současné době nejsilnější ConTEXt. Pro tvorbu rozsáhlých komplexních textů a tvorbu interaktivních dokumentů nemá mezi ostatními dvěma formáty vážnějšího konkurenta. Také jeho využití je podstatně jednodušší.

Abych tedy tuto část shrnul: lidé, kteří potřebují rychle napsat přijatelně vypadající text, popř. vědecký článek o matematice nebo fyzice, by měli patrně sáhnout po LATEXu, nejspíše s pomocí LYXu. Lidé, kteří si chtějí všechno naprogramovat sami, by měli sáhnout po plainu. Pro ostatní by mohl představovat rozumnou alternativu právě ConTEXt. Umožní jim snadno vytvářet velmi působivé, vzájemně graficky odlišené dokumenty a interaktivní prezentace. Také vydavatelství vědecké a počítačové literatury (ale i beletrie) by mohla shledat ConTEXt nesmírně zajímavým – zvlášť pokud potřebují paralelně vytvářet tištěné a „obrazovkové“, popř. interaktivní verze svých knih.

Odkazy

[1] Hans Hagen: ConTEXt*: the manual*. `www.pragma-ade.com`
[2] Hans Hagen: *metafun*. `www.pragma-ade.com`
[3] Pragma ADE: *Automatic tables*. `www.pragma-ade.com`
[4] Pragma ADE: *Fields, Widgets, References*. `www.pragma-ade.com`
[5] Pragma ADE: ppchTEX *examples*. `www.pragma-ade.com`

[6] Pragma ADE: *Flowcharts*. www.pragma-ade.com
[7] Pragma ADE: *Tabulation*. www.pragma-ade.com
[8] Hans Hagen: Tabulating in ConTeXt: text flow tables. *MAPS* 22/1999, s. 153–161.
[9] Ton Otten: ppchTeX: a macropackage for typesetting chemical structure formulas with TeX—release 2. *MAPS* 20/1998, s. 149–209.
[10] Pragma Ade: *XML in* ConTeXt. www.pragma-ade.com
[11] Taco Hoekwater: Comparing ConTeXt and LaTeX. *MAPS* 20/1998, s. 280–285.
[12] Hans Hagen: Pretty printing TeX, METAPOST, Perl and JavaScript. *MAPS* 20/1998, s. 286–289.
[13] Hans Hagen: The Calculator Demo: Integrating TeX, METAPOST, JavaScript and PDF. *MAPS* 20/1998, s. 290–296.
[14] Hans Hagen: The NTG MAPS bibliography from SGML to TeX to PDF. *MAPS* 23/1999, s. 32–47.
[15] Hans Hagen: TeX as presentation tool: an introduction to the ConTeXt presentation environment. *MAPS* 23/1999, s. 84–89.
[16] Hans Hagen: Typesetting Flow Charts: lets TeX and METAPOST do the job. *MAPS* 23/1999, s. 90–102.
[17] Hans Hagen: *Puzzling graphics in METAPOST*. www.pragma-ade.com
[18] Hans Hagen: *Beyond the bounds of paper but within the bounds of screens: The perfect match of TeX and Acrobat*. www.pragma-ade.com
[19] Adobe: *Acrobat Forms JavaScript Object Specification*. www.adobe.com

Summary: ConTeXt

Hardly any TeX user makes use of TeX in its "native" form as presented by the `initex` and `virtex` programs. Instead, a particular *format*—macro package, prepared by someone in order to make his or her work easier, is made use of. Nowadays there are two such formats which are mainly used in the Czech Republic: plain TeX and LaTeX. In the near future one more format could join them: ConTeXt. And not only join but maybe even displace LaTeX from its dominant position. In order to make it happen the author wishes to get you acquainted with ConTeXt in this contribution.

Zkušenosti ze sazby Zpravodaje Československého sdružení uživatelů TEXu

Zdeněk Wagner

Zpravodaj Československého sdružení uživatelů TEXu otiskuje články, které jsou psány různými autory. Každý z autorů používá své oblíbené prostředí, své balíky maker a dokument vytváří ve svém oblíbeném operačním systému. Hlavní problém není v tom, že autorovo prostředí je odlišné od systému, který je používán v redakci, ale plyne z toho, že je nutno do jednoho LATEXového dokumentu včlenit velmi různorodé články, z nichž některé jsou psány v plain TEXu. V tomto příspěvku je předvedeno, jak se při včleňování jednotlivých článků postupuje a jak se řeší konflikty mezi balíky maker.

Mimo tištěnou formu je Zpravodaj zveřejňován i ve formě elektronické a některé informace jsou umisťovány na WWW. V článku je ukázáno, jak je tato činnost automatizována.

Převzetí příspěvku

I když se autor článku snaží sebevíc, jen zcela výjimečně se stane, že jej lze zařadit do Zpravodaje bez redakčních zásahů. Redakční zásah je nemyslitelný bez toho, aby text byl správně zobrazen v editoru. Protože sazba se provádí v emTEXu v operačním systému OS/2, je prvním krokem vždy konverze do kódování CP852. Schopnost současné implementace TEXu založené na Web2c, která umožňuje zpracovat text v libovolném kódování podle přepínače uvedeného na prvním řádku souboru, nebo použití balíku INPUTENC, se pro redakční práci jeví jako vlastnost zcela zbytečná.

Později bude ukázáno, že některé balíky maker způsobují navzájem různé typy konfliktů. Každý příspěvek je tedy nejprve zpracován samostatně, aby redaktor mohl posoudit záměr autora.

Převod článku do jednotného vzhledu

Přestože styl pro psaní článků do Zpravodaje byl zveřejněn na WWW stránkách sdružení, většina příspěvků přichází v jiném formátu. U příspěvků, psaných v plain TEXu, je to celkem očekávané, neboť dodávaný styl je použitelný pouze pro LATEX 2_ε. Převod článku do jednotného vzhledu je činnost, jejíž složitost se případ od případu liší. Formát zdrojového textu vychází ze standardní třídy

ARTICLE. Základní převod LaTeXovských dokumentů, které jsou psány podle běžných zvyklostí, tedy většinou spočívá jen v úpravě formátování nadpisu. Dokonce ani převedení textu, napsaného původně pro třídu REPORT, není velkým problémem. V závislosti na hloubce členění kapitol a sekcí stačí použít následující definice:

```
\let\chapter\section
\let\section\subsection
\let\subsection\subsubsection
```

Máme-li tedy zaveden styl pro Zpravodaj, pak se na prvním řádku ztotožní makro `\chapter` s obsahem makra `\section`, na dalších řádcích se totéž provede s ostatními makry. Tímto postupem lze jít do libovolné hloubky, pořadí definic je samozřejmě důležité.

Převod příspěvků psaných v plain TeXu může být komplikovanější. Obecně však platí, že zpracování dobře strukturovaných dokumentů je i v těchto případech jednoduché. Například Petr Olšák v jednom ze svých příspěvků použil pro nadpisy sekcí následující makro:

```
\def\sub#1\par{\vskip1.5\bigskipamount\goodbreak
  \noindent{\bf #1}\par\nobreak\medskip}
```

V redakci byla využita jednoduchá úprava:

```
\def\sub #1\par{\section{#1}}
```

Po takto triviálním zásahu měly všechny nadpisy požadovaný vzhled bez nutnosti úprav uvnitř dokumentu.

Jednoduchá je též úprava tabulek vytvořených primitivem `\halign`. LaTeX je totiž akceptuje v nezměněné podobě. Jediným zásahem je doplnění obvyklého plovoucího prostředí `\begin{table}` a `\end{table}` a případně doplnění popisky s využitím `\caption`. Obdobně se zachází i s jinými definicemi – většinou fungují beze změny. Pouze je nutné dát pozor na makra, která se v LaTeXu jmenují stejně, ale jejichž význam je jiný. Typickým případem je `\line`, které se v LaTeXu využívá v prostředí `picture` pro kreslení čar, zatímco v plainu je to zkratka pro `\hbox to \hsize`.

Z hlediska prvních kroků lze tedy rozeznat v zásadě čtyři obtížné situace, s nimiž se redakce setkává:

1. Článek dodaný sedmibitově, kde písmena s diakritickými znaménky jsou zapisována pomocí TeXových sekvencí. Takový text je naprosto needitovatelný[1], ale naštěstí, od roku 1996 dosud, podobný článek přišel jen jednou. Vzhledem k tomu, že prastarý program `cstocs` v mých začátcích neexistoval jako nativní aplikace pro OS/2 a neznal kódování Windows 1250, přestal jsem jej používat a neuvědomil jsem si, že dokáže převést TeXové sekvence do osmibitového hódu s diakritikou.

[1] Lze jej sice otevřít libovolným ASCII editorem v libovolné národní verzi, ale vzhledem k počtu akcentů, vyskytujících se v běžném českém či slovenském textu, je orientace v textu značně obtížná.

2. Článek, který místo strukturních značek pro nadpisy, explicitně definuje vizuální vzhled opakovaně při každém použití. Je úplně jedno, zda je článek psán v LaTeXu nebo v plain TeXu. Pracnost úpravy je v obou případech stejná. Jen stěží lze u takových dokumentů využít automatické náhrady, nepomohou ani regulární výrazy. Celá práce se musí dělat ručně, je nepříjemná a zabere poměrně dlouhý čas. Naštěstí je úroveň autorů na dostatečné výši a nestrukturované příspěvky tvoří vzácnou výjimku.
3. Dokument připravený autorem původně pro jiný formát, většinou A4. Potíž bývá v tom, že se jen stěží daří vměstnat velké tabulky a výpisy kódu maker nebo jiných příkazů do formátu Zpravodaje. Úprava může být v některých případech velmi pracná.
4. Vzhledem k převodu Zpravodaje do PDF, o němž se zmíníme později, musí být téměř všechna makra robustní. Přitom nestačí robustnost v tom smyslu, jak ji používal LaTeX 2.09, ale je bezpodmínečně nutné, aby makra byla definována pomocí `\DeclareRobustCommand`. Bohužel některé styly nejen že toto nedodržují, ale dokonce pouhým `\def` předefinovávají standardní loga (včetně `\TeX` a `\LaTeX`). Příkladem je např. TEXNAMES.STY, který je využíván stylem pro sazbu dokumentace TeXLive. Záludnost spočívá především v tom, že chyby tímto způsobené se mohou projevit až po sestavení celého čísla, a to v jiném článku.

Tím se dostáváme k závěru druhého kroku. Zpracujeme článek s využitím stylu pro Zpravodaj samostatně. Kromě původního tvaru, v němž byl článek dodán autorem, získáme tedy další DVI-soubor, nyní již v požadovaném formátu. Není-li mezi oběma tvary zásadní odlišnost, můžeme přistoupit k dalšímu kroku.

Sestavení Zpravodaje

Sestavení příspěvků tak, aby vytvořily dané číslo Zpravodaje, se může zdát jednoduchou záležitostí. To by však bylo pravdou, kdyby každý článek začínal na nové stránce (jedno zda sudé či liché). Pak by se každý článek mohl vysadit samostatně s příslušně nastaveným počátečním číslem stránky a Zpravodaj by se složil pomocí programu, který dokáže spojovat DVI nebo postscriptové soubory. Články ve Zpravodaji však na sebe navazují uprostřed stránky, takže takový přístup není vhodný. Všechny články se tedy makrem `\input` načítají do jednoho dokumentu a sázejí se společně.

Prvním problémem je optimalizace stránkových zlomů, zejména s ohledem na rozměrné tabulky a obrázky. Některé články nedopadnou dobře, pokud začínají v nevhodném místě stránky. Proto se pořadí článků často určuje podle toho, jak budou na sebe nejlépe navazovat.

Každý článek je zpracováván ve svém vlastním prostředí, takže definice maker jsou lokální a nehrozí riziko, že by se ovlivňovaly navzájem. Někdy se však určitá

lokální makra vyskytují v názvu článku nebo v názvech kapitol. Pak je buď nutno příslušná makra definovat globálně pro celý Zpravodaj nebo upravit názvy. Bez tohoto zásahu by totiž nebylo možno vysázet obsah Zpravodaje makrem `\tableofcontents` ani převést Zpravodaj do PDF (nevytvořily by se záložky – outlines).

Větším problémem jsou konflikty mezi různými styly. Ty se totiž musí načítat v preambuli dokumentu. Pokud nějaký článek potřebuje určitý styl, musíme jej zpřístupnit pro celé číslo Zpravodaje. Mnohé programovací jazyky, zejména objektově orientované, které počítají s modularitou programů, umožňují oddělení jmenných prostorů. TEX takovou vlastnost nemá. Primitiv `\def` bez varování přepíše starou definici. LATEXovské makro `\newcommand` má alespoň výhodu v tom, že vypíše varování a původní definici zachová. Přesto nemusí být náprava vždycky jednoduchá. Předveďme si to na konfliktních balících PATH.STY a URL.STY. Oba slouží k sazbě URL a oba definují makro `\path`. Balík PATH.STY jej vša definuje tak, aby jeho použití bylo analogické makru `\verb`, zatímco v URL.STY je makro definováno tak, aby mělo jeden parametr uzavřený do složených závorek. Mohli bychom tedy použít následující sekvenci příkazů:

```
\usepackage{url}
\let\UrlPath\path
\usepackage{path}
\let\PathPath\path
```

V dokumentu bychom pak používali buď `\UrlPath` nebo `\PathPath`. Kromě toho, že bychom museli zasahovat do textu článku, má toto řešení další nevýhodu. Použijeme-li makro `\url` definované v URL.STY, TEX nahlásí chybu, že použití makra `\p@@th` neodpovídá jeho definici, případně že odstavec skončil před koncem parametru. Příčina spočívá v tom, že makro `\url` při svém rozvoji volá `\path`. Toto makro nám však předefinoval PATH.STY a při zpracování se odlišná syntaxe projeví až v makru `\p@@th`. Pokud bychom pořadí načítání balíků otočili, došlo by k analogickým chybám při zpracování jiných maker. V redakci byl problém vyřešen úpravou balíku URL.STY. Makro `\path` bylo v definici i ve všech použitích nahrazeno makrem `\Path`. Naštěstí se častěji v dokumentech používá `\url` nebo je `\path` součástí jiných uživatelských maker, takže požadované úpravy nejsou nijak drastické a PATH.STY, který je vyžadován stylem pro Zpravodaj, může s modifikovaným URL.STY koexistovat. Pochopitelně je upravený styl vložen pouze do adresáře Zpravodaje, aby příslušné číslo mohlo být vysázeno, ale nesmí se dostat do standardní prohledávací cesty, kde by mohl zbořit jiné dokumenty.

Daleko záludnější jsou konflikty ve vnitřních makrech, jež neslouží přímo uživatelům. Doporučuje se, aby jména interních maker začínala zkratkou jména balíku. Tak např. interní makra balíku GRAPHICX.STY začínají řetězcem `\Gin`, v balíku IFTHEN.STY je to `\TE@`, balík LONGTABLE.STY používá `\LT@` a MINITOC uvozuje jména svých maker řetězcem `\mtc@`. Tento zvyk ovšem není dodržován

obecně. Největší riziko konfliktů se bohužel vyskytuje u balíků, kde je dokumentace kódu stručná, nebo chybí úplně. Hledání a odstranění příčiny pak může být velmi náročné. Při sazbě Zpravodaje 1–3/2000 dokonce jeden konflikt zůstal nevyřešen. Zpravodaj byl již zkompletován a stačilo jen doplnit obálku. Jako ilustrace byl použit chemický strukturní vzorec, k jehož sazbě byl použit XeTeX verze 1.00[2]. Po doplnění obálky se celá sazba naprosto rozpadla. Nejviditelnější byly změny v prostředí `picture`, ale chyby se projevily i jinde. Nakonec byl Zpravodaj vysázen bez obálky, která byla vytvořena v dalším samostatném dokumentu. Adobe Acrobat Exchange naštěstí umí spojovat PDF soubory, takže se toto číslo na WWW bez problému dostane.

Sestavování Zpravodaje proto musí být podřízeno jasnému plánu. Každý článek máme z předchozího kroku zpracován samostatně. Nyní je vkládáme postupně do společného dokumentu. Bez ohledu na konečné pořadí článků v časopisu začneme těmi příspěvky, které buď nepoužívají žádné přidané styly, nebo jen ty styly, jejichž nekonfliktnost již byla otestována (např. při sazbě předchozích čísel). Jestliže nově přidaný článek způsobí problémy v již zařazených článcích, musíme jej vyřadit a pak vyřazujeme postupně jednotlivé balíky. Tímto postupem najdeme balík, který konflikt způsobuje. Pak je nutno objevit metodu, jak tento konflikt odstranit. Obecný postup neexistuje. Musíme se vybavit znalostmi TeXu a trpělivostí. Navíc může být problém úplně jinde, než se původně zdá, nebo může být zamaskován. Např. dokumentace k TeXLive používá svůj vlastní balík TEX-LIVE.STY, který však načítá další balíky. Balík TEX-LIVE.STY sám o sobě konfliktní není. Načítá však dva balíky, které konflikt se stylem pro Zpravodaj vyvolají. Jsou to již zmíněné TEXNAMES.STY a URL.STY.

Elektronické verze

Zpravodaj vychází s dvanáctiměsíčním zpožděním ve formátu PDF, na některých CD se objevuje se zpožděním menším. K vytvoření PDF je využíván balík PDFHDR.STY, který byl popsán ve Zpravodaji č. 1–2/1999. S jeho pomocí jsou generována postscriptová makra, jimž rozumí komerční Adobe Acrobat Distiller. Převod vlastního textu je zcela bezproblémový. Potíže se vyskytnou jen v záložkách (outlines, někdy zvaných též bookmarks). Mechanismus jejich tvorby modifikovaným makrem `\tableofcontents` je popsán ve výše zmíněném článku. Příčiny potíží jsou dvojí: nedostatečně robustní makra (definovaná jinak než prostřednictvím `\DeclareRobustCommand`), která většinou způsobí přeplnění paměti, a makra, která ve své expanzi obsahují TeXové primitivy. Při vývoji balíku PDFHDR.STY na to bylo pamatováno a řada potenciálních zdrojů takových problémů byla ošetřena. Pochopitelně se na mnohá makra zapomnělo. Proto seznam

[2]Novější verze dosud nepronikla na CD TeXLive 5 ani na CD CTAN a vzhledem ke spěchu nebylo možno hledat již zveřejněnou verzi 2.00 na Internetu.

maker, která lze považovat za standardní a měla by ve stylu PDFHDR.STY být ošetřena, neustále narůstá. Zbývající makra se pak vkládají v daném čísle Zpravodaje do pomocného makra `\pdf@hook`.

Okamžitě po dokončení sazby je obsah Zpravodaje zveřejňován na WWW. Text odpovídající stránky je vytvářen upraveným makrem `\tableofcontents`. Některé informace, které by WWW stránka měla obsahovat, se však ve Zpravodaji nevyskytují. Výsledek se tudíž musí nepatrně editovat. Proto máme přepínač `\if@html`, který zajistí, že si nepřepíšeme již ručně dokončenou stránku.

Při zápisu HTML se projeví stejná potíž jako při tvorbě záložek v PDF. Navíc nám vadí makra, která slouží právě pro tvorbu záložek a hypertextových odkazů z obsahu. Úvodní nastavení je proto následující:

```
\def\html@setup{\let\protect\noexpand \pdf@setup
  \csname pdf@hook\endcsname
  \def\pdftoclink##1##2{##2}\def\pdfannot##1##2##3{##3}%
  \let\PDFannot\pdfannot}
```

Pokud vytváříme HTML, musíme upravit makra pro sazbu obsahu. Provedeme to tak, že uschováme původní definice a vytvoříme nové, které nejprve provedou zápis do WWW stránky a pak zavolají původní makro.

```
\if@html
\def\html@sections{%
  \let\zw@l@section\l@section
  \let\zw@l@subsection\l@subsection
  \def\l@section##1{\begingroup \html@setup
    \immediate\write\html@bsah{<b>##1</b><p>}\endgroup
    \zw@l@section{##1}}%
  \def\l@subsection##1{\begingroup \html@setup
    \immediate\write\html@bsah{##1<p>}\endgroup
    \zw@l@subsection{##1}}%
}
\fi
```

V následujícím kroku připravíme soubor pro zápis HTML stránky a definujeme makra `\cl@nek` a `\l@clanek`, jež zapisují do obsahu autory a názvy článků. Podmíněný příkaz `\@ifundefined` zajistí, že se v obsahu v PDF nevytvoří hypertextový odkaz, pokud stránka s návěštím neexistuje. Taková situace sice nastane jen v případě, že se přehlédne nějaká chyba, ale nezobrazený rámeček v obsahu je dobrou indikací, že není vše v pořádku.

```
\newwrite\html@bsah
\DeclareRobustCommand\cl@nek[2]{#1: #2}
\def\l@clanek#1#2{\if@html \begingroup \html@setup
  \immediate\write\html@bsah{#1<p>}\endgroup \fi
  \@ifundefined{PDF@#2@}% if PDF destination undefined
    {\@dottedtocline{-2}{\z@}{2em}{#1}{#2}}% undefined
```

```
      {\@dottedtocline{-2}{\z@}{2em}{#1}{%
           \pdflink{\@nameuse{PDF@#2@}}{#2}}}% defined
   }

\def\l@clanek#1#2{\if@html \begingroup \html@setup
  \immediate\write\html@bsah{#1<p>}\endgroup \fi
  \@dottedtocline{-2}{\z@}{2em}{#1}{#2}}
```

Soubor pro zápis je otevřen až při provádění makra `\tableofcontents`. V něm je současně řízeno i vytváření záložek pro PDF. To se zapíná makrem `\PDFlisting`. Všimněte si, že se toto makro volá zdánlivě těžkopádným rozvojem s využitím primitivu `\csname`. Pokud totiž toto makro použije uživatel, který nemá instalován balík PDFHDR.STY, pak jej TEX nahradí primitivem `\relax` a k žádné chybě při zpracování nedojde. Ostatní makra, která souvisí s tvorbou PDF, jsou pro takový případ vhodně předefinována na jiných místech stylu pro Zpravodaj.

```
{\catcode`\_=12
\gdef\U@score{_}}

\def\tableofcontents{\csname PDFlisting\endcsname
\begingroup
  \if@html
  \def\az{-}
  \edef\CS@cislo@rok{\the\cislo/\the\rok}
  \def\MakePDFcislo 20##1.{\edef\cislopdf
                           {pdf/bul\U@score ##1.pdf}}
  \expandafter\MakePDFcislo\the\rok\the\cislo.
  \immediate\openout\html@bsah bul\the\rok\the\cislo.shtml
  \immediate\write\html@bsah
    {<!--\hash set var="title"
      value="Obsah Zpravodaje č. \CS@cislo@rok"}
  \immediate\write\html@bsah
    {--><!--\hash include virtual="head.shtml" -->}
  \fi
  \parindent \z@ \parskip 6pt
  {\bfseries \fontsize{14.4}{15dd}\selectfont OBSAH\par}%
  \ifPDF
    \addtocontents{bkm}{\protect\pdfline
        {\pdf@clanek}{OBSAH}{\thepage}}%
  \fi
  \vskip 15dd \toc@nl
  \@starttoc{toc}\par
```

```
    \if@html
    \immediate\write\html@bsah{<hr>}
    \immediate\write\html@bsah
      {<!-- P><a href="\cislopdf">Elektronická verze</a>
                                (??? KB) -->}
    \immediate\write\html@bsah
      {<!-- P>Další číslo <a href="bul.shtml">?/????</a -->}
    \immediate\write\html@bsah
      {<P>Minulé číslo <a href="bul.shtml">?/????</a>}
    \immediate\write\html@bsah
      {<!--\hash include virtual="adresa.shtml" -->}
    \immediate\write\html@bsah{</BODY>}
    \immediate\write\html@bsah{</HTML>}
    \immediate\closeout\html@bsah \fi
  \endgroup \thispagestyle{empty}%
  \vskip 15dd\relax}
```

WWW stránky Zpravodaje jsou založeny na využití SSI (Server Side Includes). Pomocí příkazů, které vypadají jako SGML komentáře, lze snadno nadefinovat proměnnou, obsahující titul WWW stránky, a načíst standardní hlavičku a zakončení. Lze tedy jednoduše zachovat a případně i změnit jednotný vzhled všech stránek. Místo některých informací se doplňují otazníky. Zejména údaje o následujícím čísle jsou neznámé. Ty se pak doplňují dodatečně. Například WWW stránka s obsahem Zpravodaje č. 1–3/200 vypadá v době přípravy tohoto textu následovně (automaticky generované řádky jsou dlouhé, pro zveřejnění v tomto dokumentu byly ručně přelámány):

```
<!--#set var="title" value="Obsah Zpravodaje č. 1-3/2000"
--><!--#include virtual="head.shtml" -->
Jiří Veselý: Dekadická soustava -- úvodní poznámka<p>
Sebastian Rahtz, Michel Goossens: Príručka TeX Live,
  piate vydanie<p>
Jiří Tesař: DVIBRAILLE<p>
David Antoš: TeX Versioning System aneb jak všechny
  zdrojové soubory uložiti<p>
Zdeněk Wagner: Spolupráce databáze s LaTeXem<p>
Tomáš Hála: Jedna z možností databázového publikování
  v LaTeXu<p>
Josef Svoboda: Dojmy z 19. bienále grafického designu
  v Brně<p>
Zuzana Došlá, Roman Plch a Petr Sojka: CD-ROM
  Matematická analýza s programem Maple:
  1. Diferenciální počet funkcí
```

```
98    více proměnných<p>
99  Karel Skoupý: Zpráva o prezentaci NTS na TUG 2000.<p>
100 <i>Toto číslo obsahuje CD TeX Live 5</i><P>
101 <hr>
102 <!-- P><a href="pdf/bul_0013.pdf">Elektronická verze</a>
103   (724 KB) -->
104 <!-- P>Další číslo <a href="bul.shtml">?/????</a -->
105 <P>Minulé číslo <a href="bul994.shtml">4/1999</a>
106 <!--#include virtual="adresa.shtml" -->
107 </BODY>
108 </HTML>
```

Přechod k novému designu WWW stránek, k němuž došlo na počátku roku 2001, tedy vyžadoval pouze změny v souborech `head.shtml` a `adresa.shtml`, ostatní soubory zůstaly beze změny.

Závěr

Zpravodaj Československého sdružení uživatelů TEXu je specifický tím, že se zabývá právě TEXem. Riziko konfliktů mezi balíky maker, potřebnými pro sazbu jednotlivých článků, je proto vyšší než v případě jiných sborníků. Na základě zkušeností, které vyplynuly z několikaleté praxe, lze doporučit pro editory jiných sborníků a časopisů jedno základní pravidlo. Pokud je pro sborník či časopis použit specifický styl, pak by měl co nejvíce vycházet ze standardních tříd ARTICLE nebo REPORT. Nejen že se takový styl autoři snadno naučí používat, ale i v případě, že jej autoři nepoužijí a dodají běžný dobře strukturovaný text, nevyžaduje zařazení takového dokumentu žádnou zvláštní námahu. Pokud je však doporučený styl velmi komplikovaný a odlišný od běžných standardů, pak to některé autory odradí natolik, že článek nenapíšou vůbec, a jiní jej dodají ve tvaru, který redaktorům způsobí několik šedivých vlasů. Jistému úsilí, potřebnému k vytvoření sborníku nebo časopisu, se však nevyhneme nikdy.

Summary: Experience from typesetting the Bulletin of the Czechoslovak TEX Users Group

The Bulletin of the Czechoslovak TEX Users Group presents articles written by different authors. Each author uses his or her favourite environment, his or her macro packages and prepares the document in his or her favourite operating system. The main problem is not caused by the fact that the author's environment differs from the system used by the editor but stems from the fact that heterogenous articles, some of which are written in plain TEX, must be embodied into a

single LaTeX document. This contribution describes the course of operation and explains the way how conflicts between macro packages are solved.

In addition to the printed form the Bulletin is also published in an electronic form and some pieces of information are presented on WWW. The article shows how this operation is automated.

Lehký úvod do XML

Jiří Kosek

Příspěvek seznámí čtenáře s jazykem XML, který přináší mnoho revolučních změn do oblasti elektronického publikování, výměny a sdílení dat a elektronického obchodu. Kromě základních principů XML se příspěvek zmíní i o souvislosti s dalšími navazujícími technologiemi (stylové jazyky, jazyky pro definici struktury dokumentu, dotazovací jazyky, jazyky pro tvorbu odkazů).

Jazyk XML (eXtensible Markup Language) je poměrně nový značkovací jazyk. Mezi jeho největší výhody patří naprostá otevřenost a velká flexibilita. Díky tomu se během krátké doby stalo XML velice populární. XML vzniklo zjednodušením jazyka SGML (Standard Generalized Markup Language), který je ISO normou 8879 z roku 1986. Kvůli své složitosti bylo SGML nasazováno jen ve větších aplikacích. XML je oproti tomu jednoduchý jazyk, který vytvořilo konsorcium W3C.

Úvod

Málokterá technologie se rozšířila tak rychle jako XML. Před třemi lety o ní skoro nikdo nic nevěděl, a dnes se přitom používá v mnoha aplikacích. Budeme-li se držet přesné definice zjistíme, že XML (eXtensible Markup Language) je jednoduchý rozšiřitelný značkovací jazyk. Co si pod touto definicí představíme záleží zejména na naší fantazii. V následujícím příspěvku se proto pokusím vysvětlit, co je to XML a k čemu se dá použít.

Samotný pojem XML se dnes používá ve třech trošku odlišných významech, na které se postupně podíváme. XML je

- formát pro výměnu a ukládání dat;
- metajazyk pro definici dalších jazyků;
- celá sada technologií, které úzce souvisejí s jazykem XML (XSL, XLink, XPointer, ...).

XML jako formát pro výměnu dat

Svět se začíná globalizovat, informace nabývají na důležitosti a vzrůstá potřeba jejich efektivního zpracování a vyměňování. Formátů pro výměnu dat existují stovky, ale většina z nich je jen úzce zaměřena a má mnohá omezení. XML oproti většině jiných formátu přináší mnohá vylepšení.

První přínos XML spočívá v usnadnění *vyhledávání informací*. Většina dnes dostupných informací je vytvářena a ukládána v nestrukturované podobě – jako textové soubory, webové stránky apod. Efektivní vyhledávání v takovýchto datech je podmíněno porozuměním uložené informaci. To je bohužel v dnešní době stále nevyřešený problém. Snadné je vyhledávání naopak v databázích – v nich jsou všechny údaje přehledně strukturovány. Problém je však v tom, že databáze obsahují jen nepatrný zlomek informací, které máme k dispozici.

XML přináší možnost strukturování libovolných dokumentů, včetně těch textově orientovaných. XML je značkovací jazyk, což znamená, že jednotlivé části dokumentu označujeme značkami, které přesně specifikují jejich význam. Část webové stránky internetového obchodu s knihami by proto mohla v XML vypadat třeba takto:

```
<nabídka>
  <název>The Art of Computer Programming, Vol. 1</název>
  <autor>Donald E. Knuth</autor>
  <cena>40 GBP</cena>
</nabídka>
```

Vidíme, že v XML dokumentech se podobně jako v HTML používají značky pro označení částí dokumentu. Na rozdíl od HTML si však můžeme volit vlastní názvy značek, a tak co nejpřesněji vyznačit význam jednotlivých informací v textu.

Takto strukturované informace lze velice snadno prohledávat. Průměrně zdatný programátor by dokázal za pár hodin napsat program, který po zadání dotazu typu „Najdi mi stránku, kde se dá nejlevněji koupit kniha The Art of Computer Programming“ skutečně nalezne požadovanou informaci. Stačí nalézt všechny stránky, kde je v tagu `<nabídka>` obsažen tag `<název>`, který obsahuje hledaný text, a z těchto stránek vybrat tu, kde je v tagu `<cena>` uvedena nejnižší hodnota.

První z výhod, kterou XML přináší, je tedy usnadnění vyhledávání především v rozsáhlých kolekcích dokumentů jako je např. Web. Vyžaduje to samozřejmě, aby všichni autoři stránek označili důležité informace odpovídajícími značkami. Navíc by se měly pro stejné věci používat značky se stejnými názvy. Když si každý vymyslí vlastní názvy značek, bude mít sice pocit svobody, ale situaci tím nijak nepomůže. Dále v přednášce se proto podíváme na to, jak lze formálně specifikovat množinu značek, které lze používat v XML dokumentu.

Mezi další velkou výhodu XML patří jeho *univerzálnost*. V dokumetech lze používat libovolné značky, lze je do sebe zanořovat. Do XML dokumentu tak lze velice přirozeným způsobem uložit téměř libovolnou informaci. XML formát si poradí jak s textově orientovanými daty (kniha, článek, webová stránka), tak i s databázovými údaji (ceník, tabulka zaměstnanců apod.).

Univerzálnost je podpořena i velice dobrou *mezinárodní podporou*. XML již od samého počátku počítá s tím, že existují i jiné jazyky než angličtina. Jako standardní znaková sada se používá 32bitové ISO 10646. V jednom dokumentu tak můžeme míchat dohromady všechny dnes na naší planetě běžně používané znaky. Nic nám zároveň nebrání v použití libovolného kódování, které nám vyhovuje. V Česku se jedná zejména o kódování ISO 8859-2 a windows-1250. V každém XML dokumentu se standardním způsobem zaznamenává informace o použitém kódování.

Velmi důležitou věcí, která by si možná zasloužila první místo v našem výčtu, je *otevřenost*. Formát XML není žádný proprietární formát nějaké komerční firmy. Specifikace XML [XMLSPEC] je poměrně jednoduchá a krátká, a kdokoliv si ji může zdarma stáhnout ze stránek konsorcia W3C [W3C].

Otevřenost a univerzálnost formátu je velice důležitou vlastností, zvláště pokud nám záleží na námi vytvořených datech. Při použití XML nejsme omezeni na používání proprietárních aplikací. Můžeme používat různé nástroje od různých firem na různých platformách, nemusíme se bát, že za pár let si v nové verzi editoru nepřečteme staré dokumenty. Je mnoho oblastí, kde jsou tyto vlastnosti klíčové – například dokumentace k různým zařízením – takové letadlo, raketové silo nebo soustruh mají životnost mnohonásobně delší než verze x.y nějakého dnes běžně používaného textového procesoru. Použitím XML dáme najevo, že důležitá jsou naše data, a ne aplikace, které pro práci s nimi používáme.

Díky bohatému označkování informací můžeme XML dokumenty velice snadno konvertovat do dalších formátů, můžeme opakovaně využívat již jednou existující informace. Tato činnost je v mnoha případech dokonce nezbytná. XML dokumenty obsahují informace, ale nijak nedefinují jejich vzhled. Pro člověka je však nutné údaje nějak přehledně zformátovat. Pro tyto účely existují speciální *stylové jazyky*, které umožňují přímé zobrazení XML dokumentu, nebo jeho převod do dalších formátu jako je HTML, XHTML, PDF apod.

Základy syntaxe XML

XML patří mezi značkovací jazyky (markup languages). Důležité části dokumentu se označují pomocí značek. V terminologii XML se jednotlivým označeným částem dokumentu říká *elementy*. Elementy do sebe mohou být navzájem vnořené a tím dle potřeby zachycovat strukturu informací uložených v dokumentu.

Kdybychom například do XML ukládali elektronickou podobu knihy, skládal by se dokument z elementů kapitola. Každý element kapitola by pak obsahoval element nadpis a několik elementů pro odstavce. Příkladem z odlišné oblasti je uložení databázové tabulky do XML dokumentu. Dokument bude obsahovat několik elementů odpovídajících jednotlivým záznamům (řádkám) tabulky. Každý z těchto elementů pak samozřejmě bude obsahovat další elementy pro jednotlivé položky tabulky. Každý XML dokument se dělí postupně na menší a menší části.

Elementy

Elementy se v textu vyznačují pomocí tzv. *tagů*. Většině elementů odpovídají dva tagy – počáteční a ukončovací.

```
<para>Toto je obsah elementu para.</para>
```

Naše ukázka obsahuje jeden element `<para>`. Jeho obsah je vyznačen pomocí tagů `<para>` (počáteční tag) a (ukončovací tag). Jen na okraj poznamenejme, že výše uvedená ukázka je asi nejjednodušší XML dokument, který vůbec můžeme vytvořit.

Názvy tagů se zapisují mezi znaky '`<`' a '`>`'. Ukončovací tag má před svým názvem ještě znak '`/`', aby se snadno odlišil od počátečního.

Některé elementy nemusejí mít žádný obsah. Můžeme je samozřejmě zapisovat tak, že za počátečním tagem uvedeme hned ten ukončovací.

```
<para>Toto je obsah elementu para.<br></br>
A tohle taky.</para>
```

Není to však příliš pohodlné, a proto můžeme v XML použít ještě jednu variantu tagu, která říká, že element nemá žádný obsah. Za jméno elementu v počátečním tagu uvedeme znak '`/`'. Ukončovací tag se pak už nepoužije.

```
<para>Toto je obsah elementu para.<br/>
A tohle taky.</para>
```

Každý XML dokument musí obsahovat pro všechny počáteční tagy odpovídající ukončovací tag, nebo musí být počáteční tag zapsán jako element s prázdným obsahem. To je velký rozdíl oproti jazyku HTML, kde v mnoha případech můžeme ukončovací tagy vynechat. Při návrhu XML byla jedním z požadavků snadná implementace parserů, které budou XML dokumenty načítat. Tomu odpovídá i větší striktnost syntaxe XML oproti HTML.

Atributy

Elementy jsou základním stavebním kamenem každého dokumentu. U každého počátečního tagu můžeme použít ještě *atributy*. Atributy se používají k upřesnění významu elementu, k přidání dalších důležitých informací.

```
<para zabezpečení="důvěrné">Nějaká tajná informace.</para>
```

V naší ukázce jsme atributu `zabezpečení` přiřadili hodnotu `důvěrné`. Hodnotu atributu musíme vždy uzavřít do uvozovek nebo do apostrofů. U jednoho tagu lze použít více atributů najednou, stačí je oddělit mezerou.

```
<para zabezpečení="důvěrné" autor="Jan Novák">Nějaká
tajná informace.</para>
```

Znakové entity

Vzhledem k tomu, že se znak '`<`' používají k zahájení tagu, není možné ho zapsat do dokumentu jen tak. Pro jeho zápis musíme použít tzv. *znakovou entitu*. Pro zápis znaku '`<`' je určena entita `<`. Existuje i entita `>` pro zápis '`>`', ale její používání není nutné.

```
Vyřešte nerovnost 3x &lt; 5
```

Pro samotný zápis ampersandu (`&`) se používá znaková entita `&`.

```
Křupavé rohlíčky vám dodá pekařství Žemlička &amp; syn
```

Pokud potřebujeme uvnitř hodnoty atributu použít zároveň uvozovky i apostrofy, s výhodou využijeme odpovídající entity `&quot;` a `&apos;`.

```
<monitor úhlopříčka="15&quot;"/>
```

Použití entit `&quot;` a `&apos;` se někdy můžeme vyhnout použitím apostrofů pro oddělení obsahu atributu:

```
<monitor úhlopříčka='15"'/>
```

Kořenový element

Každý XML dokument musí být celý obsažen v jednom elementu. Následující ukázka tedy není správný XML dokument, protože se skládá z několika samostatných elementů.

```
<nadpis>Pokusný nadpis</nadpis>
<odstavec>První odstavec</odstavec>
<odstavec>Druhý odstavec</odstavec>
<odstavec>Třetí odstavec</odstavec>
```

Stačí však přidat *kořenový element*, který vše „obalí“, a dokument je rázem v pořádku.

```
<článek>
  <nadpis>Pokusný nadpis</nadpis>
  <odstavec>První odstavec</odstavec>
  <odstavec>Druhý odstavec</odstavec>
  <odstavec>Třetí odstavec</odstavec>
</článek>
```

Kódování znaků

Jako znaková sada se v XML dokumentech používá ISO 10646. Tato znaková sada je 32bitová, takže obsahuje dostatek pozic pro všechny znaky všech abeced používaných na Zemi. V současné době je definováno 49 194 znaků, jejichž kódy jsou shodné s Unicode 3.0.

Do dokumentu se znaky musí zapisovat pomocí určitého kódování, které definuje, jak se kód znaku bude reprezentovat nějakou sekvencí bajtů. Standard XML vyžaduje, aby všechny aplikace podporovaly alespoň kódování UTF-8 a UTF-16.

UTF-8 kóduje jeden znak do různého počtu bajtů. Znaky anglické abecedy jsou uloženy do jednoho bajtu a jejich kód odpovídá ASCII kódu. Ostatní znaky jsou kódovány do dvou až šesti bajtů. Konkrétně české znaky s diakritikou jsou kódovány do dvou bajtů. Pokud dokument v UTF-8 otevřeme v editoru, který toto kódování nepodporuje, uvidíme místo českých znaků dost podivné dvojice znaků.

Dalším použitelným kódováním je UTF-16. Je to 16bitové kódování, jeden znak je uložen ve dvou bajtech, které přímo obsahují kód znaku.

Použití UTF-8 a UTF-16 pro české (resp. slovenské) texty není moc pohodlné. Jednak je k dispozici málo editorů, které by umožňovaly bezproblémové použití těchto kódování. Druhý problém, i když už ne tak palčivý, je zbytečné zvětšení velikosti dokumentů. V Česku se dnes používají pro české texty dvě kódování – ISO 8859-2 a windows-1250. Můžeme je použít i v XML dokumentech, ale v tomto případě musíme vždy na začátku dokumentu použít *XML deklaraci* a v ní určit kódování.

```
<?xml version="1.0" encoding="iso-8859-2"?>
```

resp.

```
<?xml version="1.0" encoding="windows-1250"?>
```

Na tuto deklaraci nesmíme zapomínat, její vynechání a použití nestandardního kódování vede často k tomu, že máme problémy vytvořit i jednoduchý korektní XML dokument.

Zobrazení XML dokumentu a kontrola syntaxe

Splňuje-li dokument všechna výše uvedená pravidla, je syntakticky v pořádku a říkáme o něm, že je *správně strukturovaný (well-formed)*. Takový dokument pak můžeme zpracovat mnoha aplikacemi, které podporují formát XML.

Úplně základní aplikací pro zpracování XML je *parser*. Parser umí číst XML dokument a kontrolovat jeho syntaxi. Parser je obvykle integrální součástí nějaké další aplikace, např. prohlížeče, který pomocí něj čte XML dokument. Asi nejjednodušším způsobem, jak zkontrolovat správnou syntaxi XML dokumentu, je otevřít jej v prohlížeči s podporou XML. Pokud je dokument v pořádku, zobrazí

se. Obsahuje-li dokument chyby, prohlížeč nás na ně upozorní. V současné době podporují XML například prohlížeče Mozilla a Internet Explorer 5.

Při zobrazování XML dokumentu prohlížeče neví, jak si přejeme jednotlivé elementy zobrazit. To lze určit pomocí stylu, který definuje způsob zobrazení. Bez něj nám Mozilla dokument zobrazí jako jeden dlouhý odstavec. Internet Explorer zobrazí zdrojový kód XML se zvýrazněnou syntaxí. K možnostem tvorby a použití stylů se v článku ještě vrátíme.

XML jako metajazyk pro definici dalších jazyků

XML umožňuje zcela libovolně volit názvy tagů. Na druhou stranu příliš volnosti škodí. Standard XML proto přímo v sobě obsahuje nástroj, který umožňuje definovat elementy přípustné v daném druhu dokumentů, jejich vzájemné vztahy a atributy. Tímto nástrojem je *DTD (Definice Typu Dokumentu)*. Vytvořením vlastního DTD vytvoříme nový jazyk, který základní charakteristiky a syntaxi přebírá z XML, ale má přesně definovanou množinu použitelných elementů.

Dnes existují stovky a možná i tisíce DTD, každé z nich definuje nový jazyk, nový formát, který je založený na XML. Mezi nejznámější jazyky tímto způsobem „odvozené“ od XML patří například:

- XHTML – nástupce jazyka HTML, plně přebírá jeho sémantiku, ale syntaxe je přizpůsobena XML.
- WML (Wireless Markup Language) – jazyk pro tvorbu jednoduchých webových stránek používaných v mobilních telefonech.
- MathML (Mathematical Markup Language) – jazyk pro zápis matematických výrazů.
- SVG (Scalable Vector Graphics) – jazyk pro 2D vektorovou grafiku, navržený speciálně pro potřeby Webu.
- DocBook – de facto standard pro tvorbu dokumentace (používá se například v dokumentačním projektu pro operační systém Linux).

Libovolný dokument můžeme pomocí parseru kontrolovat oproti DTD. Dokument, který splňuje omezení definovaná v DTD, se nazývá *validní*. Parser, který je schopen provádět validaci, nám může ušetřit mnoho práce. Kdybychom měli v XML například uložené faktury, může parser ve spojení s příslušným DTD zcela automaticky zkontrolovat, zda faktura obsahuje údaje o odběrateli, dodavateli a jednotlivé položky. Když budeme mít v XML uložen text knihy, může nám parser zkontrolovat, jestli má každá kapitola název apod.

Kontrolování validity pomocí DTD je výhodné z několika důvodů. Když od někoho naše aplikace obdrží data v XML, může mnoho kontrol provést automaticky parser. Nemusíme ručně psát mnohdy poměrně zdlouhavý a nezáživný kód ošetřující chyby ve vstupních datech.

DTD může využívat i XML editor, který autorovi dokumentu průběžně nabízí vložení jen těch elementů, které jsou v daném kontextu platné.

Možnosti DTD jsou pro velký okruh aplikací zcela dostačující, ale rozhodně neřeší zdaleka všechny problémy spojené s kontrolou syntaxe a obsahu dokumentu. V současné době konsorcium W3C dokončuje standard *XML schémata* [XSDSPEC]. Princip jejich použití je stejný jako u DTD. Oproti DTD přinášejí mnohá vylepšení, která naleznou uplatnění zejména v aplikacích, které používají XML pro ukládání hodně strukturovaných dat databázového typu.

Nejvýraznějším rysem XML schémat je bohatá podpora datových typů. U každého atributu a elementu můžeme určit přesně jeho datový typ (číslo, řetězec, datum apod.) včetně různých integritních omezení. Jsou zde dokonce nástroje pro definici referenční integrity podobně, jak je známe z relačních databází.

Podstatně byly rozšířeny i vyjadřovací schopnosti jazyka. K tomu přispěla i nová syntaxe, která je na rozdíl od DTD založena na XML.

X** technologie

Se samotným jazykem XML úzce souvisejí další technologie a jazyky, které jeho možnosti dále rozšiřují. Na ty nejdůležitější z nich se teď stručně podíváme.

Stylové jazyky

XML dokumenty popisují strukturu dat, ale nijak nedefinují, jak se mají uložené informace prezentovat uživateli, jak se mají formátovat. XML umožňuje důsledně oddělit *obsah* dokumentu od jeho *vzhledu*. Používá se přitom velice jednoduchá myšlenka *stylových jazyků*. Definice vzhledu dokumentu, resp. jeho jednotlivých částí se definuje pomocí nějakého speciálního jazyka v samostatném souboru, kterému se říká *styl*.

Chceme-li XML dokument zobrazit zformátovaný, aplikujeme na něj styl, který provede zformátování dokumentu do výsledné podoby. Tento krok může přitom být v mnoha případech zcela automatický. Například webový prohlížeč si stáhne XML dokument a z metainformace na jeho začátku zjistí, jaký má použít styl pro jeho formátování. Stáhne si tedy i odpovídající styl a dokument zobrazí rovnou zformátovaný.

Největší výhoda oproti klasickému přístupu při zpracování dokumentů je v tom, že k jednomu druhu dokumentů můžeme mít několik různých stylů. Podle potřeby pak můžeme z jednoho zdroje dat generovat mnoho různých podob. To je dnes velmi potřebná vlastnost. Například při tvorbě dokumentace chceme mít jeden dokument k dispozici v několika formátech – např. HTML, PDF, info apod. Každý formát má přitom specifické vlastnosti. Při použití běžných

technologií bychom museli ručně (v lepším případě poloautomaticky) udržovat několik verzí stejného dokumentu. Místo toho však můžeme dokument uložit do XML a vytvořit styl pro každý požadovaný výstupní formát. Například dokumentace k Linuxu (LDP) a mnoho dalších projektů dnes používá DTD DocBook pro tvorbu dokumentace. Z jedné předlohy se pak generuje několik výsledných formátů dokumentace.

Analogický přístup se začíná používat i na Webu. K webovým stránkám se začíná pomalu přistupovat i z jiných koncových zařízení než jsou PC – z mobilních telefonů a různých PDA. Tato zařízení mají omezené možnosti, často používají pro stránky jiný formát než HTML. Poskytování stránek v různých formátech lze přitom vyřešit velice jednoduše – informace uložíme na webový server v XML společně s několika styly pro jednotlivá koncová zařízení. Před odeslání stránky web-server automaticky zkonvertuje informaci pomocí stylu do formátu, který nejlépe vyhovuje klientskému zařízení.

Styly jsou výhodné i v případech, kdy potřebujeme generovat velké množství dokumentů se stejným vzhledem. Vzhled je definován na jednom místě ve stylu. Pokud s ním zpracováváme dokumenty s podobnou strukturou (tj. dokumenty vyhovují stejnému DTD), mají všechny jednotný grafický design. Při požadavku na změnu všech dokumentů (dokumentace ke všem produktům, všechny webové stránky) pak stačí změnit jeden styl a přegenerovat výsledné podoby dokumentů. Pro jeden dokument se tento přístup nevyplatí, ale pro větší množství dokumentů je již úspora práce zcela patrná.

Pro formátování XML dokumentů se dnes používají převážně dva stylové jazyky – kaskádové styly (CSS) a jazyk XSL.

Kaskádové styly umožňují pro jednotlivé elementy dokumentu definovat základní vizuální vlastnosti, jako výběr písma, jeho velikost, barvu, zarovnání apod. Hodí se jen pro velice jednoduché formátování. Různou úroveň podpory kaskádových stylů společně s XML dnes nabízí například prohlížeče Mozilla, Internet Explorer a Opera.

Speciálně pro potřeby XML byl vyvinut jazyk XSL (eXtensible Stylesheet Language). Ten obsahuje dvě části: transformační jazyk XSLT a jazyk pro abstraktní popis vzhledu dokumentu (tzv. formátovací objekty).

XSLT umožňuje velice jednoduše a efektivně provádět transformaci XML dokumentů do formátů XML, HTML a čistého textu. Při transformaci můžeme vybírat části vstupního dokumentu, třídit je, podmíněně zpracovávat apod. Síla XSLT je obrovská a těžko vyjádřitelná v jednom odstavci. Dnes se ponejvíce používá pro konvertování XML dokumentů do formátů jako je HTML a WML.

Formátovací objekty umožňují z elementárních bloků textu poskládat celý zformátovaný dokument. Vstupní XML dokument nejprve zpracujeme XSLT stylem, který však negeneruje ani HTML, ani WML, ale XML dokument obsahující formátovací objekty. Tento soubor pak zpracujeme procesorem, který provede konečné rozmístění objektů na stránku či obrazovku a zalomení řádků a stránek.

Tvorba odkazů
Jedním z impulzů pro vznik XML bylo vylepšení možnosti webových stránek. XML proto přináší i řádově lepší možnosti pro tvorbu hypertextových odkazů mezi dokumenty. Pro tvorbu odkazů se používají speciální jazyky XLink a XPointer.

API pro zpracování XML
Pokud chceme s XML pracovat v našich aplikacích, nemá cenu si psát vlastní parser, který umí načítat XML dokumety. Místo toho je lepší použít již hotové knihovny. Většina z nich přitom podporuje standardizovaná rozhraní pro zpracování XML. Dnes se převážně používají rozhraní DOM a SAX.

DOM (Document Object Model) modeluje XML dokument pomocí stromové hierarchie objektů v paměti. V aplikaci pak můžeme velice snadno přistupovat k libovolným částem dokumentu.

Rozhraní SAX (Simple API for XML) je řízené událostmi. Během čtení dokumentu volá parser námi definované funkce, které obsluhují důležité části dokumentu – počáteční a koncové tagy, obsah elementů apod.

Dotazovací jazyky
V současné době se velké úsilí věnuje vývoji kvalitních dotazovacích jazyků. Jediný zatím standardizovaný jazyk je XPath (XML Path Language), který umožňuje tvorbu poměrně jednoduchých dotazů, které vybírají části vstupního dokumentu. XPath se používá v několika dalších standardech (XSLT, XML schémata).

XPath je však orientován převážně na textově založené dokumenty. W3C konsorcium pracuje na jazyku XML QL, který se v sobě snaží efektivně a elegantně sloučit možnosti XPath, SQL a části XSLT.

Závěr

Před pár lety jeden z tvůrců XML prohlásil, že „XML je ASCII budoucnosti“. Já osobně mám pocit, že tato vize se již začíná naplňovat. Tento příspěvek rozhodně nemohl pokrýt celou problematiku XML a navazujících technologií. Doufám, že vám alespoň poslouží jako dobrý odrazový můstek při studiu podrobnějších zdrojů uvedených v seznamu literatury.

Odkazy

[DOM] Hors Arnaud Le, Hégaret Philippe Le, Wood Lauren, Nicol Gavin, Robie Jonathan, Champion Mike, Byrne Steve: Document Object

Model (DOM) Level 2 Core Specification [http://www.w3.org/TR/DOM-Level-2-Core/]. W3C, 2000.

[SAX] Megginson, David: SAX 2.0: The Simple API for XML [http://www.megginson.com/SAX/]. 2000.

[W3C] Stránky konsorica W3C. http://www.w3.org/

[XLINK] DeRose Steve, Maler Eve, Orchard David: XML Linking Language (XLink) [http://www.w3.org/TR/xlink]. W3C, 2000.

[XMLCS] Kosek, Jiří: XML pro každého [http://www.kosek.cz/xml/]. Grada Publishing, 2000.

[XMLSPEC] Bray Tim, Paoli Jean, Sperberg-McQueen C. M.: Extensible Markup Language (XML) 1.0 [http://www.w3.org/TR/REC-xml]. W3C, 1998.

[XPTR] DeRose Steve, Daniel Ron Jr., Maler Eve: XML Pointer Language (XPointer) [http://www.w3.org/TR/xptr]. W3C, 2001.

[XSDSPEC] Fallside, David C.: XML Schema Part 0: Primer [http://www.w3.org/TR/xmlschema-0/]. W3C, 2000.

[XSL] Adler Sharon, Berglund Anders, Caruso Jeff, Deach Stephen, Grosso Paul, Gutentag Eduardo, Milowski Alex, Pernell Scott, Richman Jeremy, Zilles Steve: Extensible Stylesheet Language (XSL) – Version 1.0 [http://www.w3.org/TR/xsl]. W3C, 2000.

[XSLT] Clark, James: XSL Transformations (XSLT) Version 1.0 [http://www.w3.org/TR/xslt]. W3C, 1999.

Summary: Gentle Introduction to XML

XML language brings many revolutional changes into the area of electronic publishing, information sharing and interchange and e-business. This article introduces basic XML principles and its relation to related technologies (stylesheet langauges, document type definition langauges, query languages and linking languages).

PassiveTEX

Jiří Kosek

PassiveTEX je sada maker, která umožňuje formátování XML dokumentů. Pro popis výsledného vzhledu dokumentu se přitom používá standardní stylový jazyk XSL (eXtensible Stylesheet Language). Tvorba stylů v XSL je pro

mnoho uživatelů snazší než přímé programování v TEXu. PassiveTEX zcela odstiňuje uživatele od makrojazyka TEXu a umožní tak využít kvalitní výstup TEXu širšímu okruhu uživatelů než dnes. Během přednášky se posluchači seznámí se základními principy XSL a shlédnou praktické ukázky použití PassiveTEXu.

Úvod

XML je formát vhodný pro ukládání mnoha různých druhů dat. Data je však nutné prezentovat uživateli, XML dokument proto musíme zformátovat. Pro formátování XML dokumentů existuje speciální jazyk XSL (eXtensible Stylesheet Language). Jazyk XSL obsahuje dvě části – transformační jazyk XSLT a formátovací objekty (FO). Transformační část jazyka umožňuje definovat transformaci do XML dokumentu s jinou strukturou i značkami. Formátovací část jazyka pak definuje speciální jazyk s XML syntaxí, který definuje jednotlivé objekty zformátovaného dokumentu a jejich vizuální vlastnosti.

Formátování XML dokumentu probíhá tedy tak, že se nejprve provede transformace do formátovacích objektů. V dalším kroku se zpracují formátovací objekty a dostaneme výsledek – ať už přímo na obrazovce nebo třeba v PDF souboru.

Programů, které umějí provádět transformace definované pomocí XSLT, je dnes již mnoho. O to větší nouze je však o programy, které umějí dokument s formátovacími objekty zalomit do výsledných stránek. Jedním z těchto programů je i PassiveTEX, který při formátování využívá schopnosti typografického systému TEX. V tomto příspěvku se nejprve podíváme na princip fungování XSL stylů a poté si ukážeme některé možnosti PassiveTEXu.

XSLT aneb není nad dobrého sluhu

Základní myšlenka XSLT je velice jednoduchá. Styl obsahuje šablony, které slouží pro zpracování určité části (většinou určitého elementu) vstupního dokumentu. XSLT procesor postupně prochází vstupní dokument, a pokud najde šablonu, která umí obsloužit danou část dokumentu, provede ji. Toto jednoduché chování lze samozřejmě ovlivňovat mnoha způsoby – části šablon lze provádět podmíněně, části vstupního dokumentu můžeme před zpracováním setřídit apod. Na podrobný popis možností XSLT zde bohužel nemáme prostor.

Styl v XSLT nepoužívá žádnou speciální syntaxi, jedná se o XML dokument. V tomto dokumentu se používají vždy dvě sady značek – instrukce pro XSLT procesor a značky, které se mají objevit ve výstupním dokumentu. Když používáme XSLT pro převod z XML do HTML, obsahuje styl XSLT instrukce a HTML

tagy. Pokud provádíme transformaci do formátovacích objektů, obsahuje XSLT styl kromě instrukcí ještě elementy formátovacích objektů. Kombinování více různých sad značek v jednom dokumentu je umožněno díky standardnímu mechanismu jmenných prostorů, který je součástí XML.

Základní princip fungování XSLT si ukážeme na jednoduchém příkladě. Dejme tomu, že máme v XML dokumentu uloženy informace o přednáškách konaných v rámci SLT (viz příklad 1). Nyní chceme vytvořit HTML stránku, kde bude program ve formě tabulky se čtyřmi sloupci, ve kterých bude postupně název přednášky, jméno autora, vysílající firma a abstrakt.

Příklad 1: XML podoba seznamu přednášek – slt.xml

```
<?xml version="1.0" encoding="iso-8859-2"?>
<prednasky>
  <akce>SLT 2001</akce>
  <prednaska>
    <nazev>Clusterová řešení na Linuxu</nazev>
    <autor>Jan Kasprzak</autor>
    <firma>FI MU Brno</firma>
    <abstrakt>Operační systém Linux je svojí modularitou a dostupností
      zdrojových textů vhodný i jako součást rozsáhlejších systémů.
      V přednášce budou popsány základní typy clusterů &#x2013;
      výpočetní clustery, load-balancing clustery a high availability
      clustery. Dále software, pomocí kterého je možno jednotlivé typy
      clusterů realizovat a některé zkušenosti s prací
      clusterů.</abstrakt>
  </prednaska>
  <prednaska>
    <nazev>BIRD Internet Routing Daemon</nazev>
    <autor>Martin Mareš</autor>
    <firma>SuSE ČR</firma>
    <abstrakt>Jádro Linuxu od nepaměti (i z pohledu kernelových
      hackerů, a to už je co říci) disponuje velice rychlou a flexibilní
      implementací TCP/IP. K tomu, aby se z Linuxu stal špičkový router
      použitelný i ve velkých sítích, ovšem chybí kvalitní daemon
      implementující dynamické routovací protokoly (BGP, OSPF, ...)
      a předávající zkonstruované routovací tabulky jádru. Před časem
      vznikl na MFF UK projekt BIRD, který se snaží tuto mezeru zaplnit
      a přinést (nejen) Linuxu i další síťové vymoženosti, ve světě
      &#x201e;opravdových&#x201c; routerů zatím velice řídké: dynamické
      rekonfigurování, více routovacích tabulek, BGP multihoming
      a programovatelnou filtraci, to vše jak pro IPv4, tak i IPv6. Více
      se dočtete na http://bird.network.cz/.</abstrakt>
  </prednaska>
  <prednaska>
    <nazev>Využití XML rozhraní při tvorbě aplikací</nazev>
    <autor>Jan Pazdziora</autor>
    <firma>FI MU Brno</firma>
    <abstrakt>Oddělení obsahu a prezentace je módní slogan, bohužel
```

```
      většina řešení skončí u některého z template
      přístupů. Prezentované řešení nad AxKitem využívá XML jako vrstvu,
      která generování dat od formátování striktně odstiňuje. Zaměříme
      se zejména na aplikace pracující s dynamickým obsahem
      a prozkoumáme, jak vypadá dobře strukturované XML a jakých rozličných
      výstupů je možno z jedněch databázových dat dosáhnout.</abstrakt>
  </prednaska>
  <prednaska>
    <nazev>IP verze 6</nazev>
    <autor>Pavel Satrapa</autor>
    <firma>TU Liberec</firma>
    <abstrakt>Základní seznamení s vlastnostmi nové verze
      IP. Adresovaní, směrovaní, bezpečnostní mechanismy, podpora
      mobilních zařízení. Aktuální stav implementací IPv6 a především
      jeho implementace v Linuxu.</abstrakt>
  </prednaska>
</prednasky>
```

Vytvořit takovou transformaci je velice jednoduché. V podstatě stačí „přejmenovat“ naše XML tagy na HTML tagy pro tvorbu tabulek. Příklad 2 obsahuje ukázku jednoduchého stylu, který můžeme použít pro převod do HTML. Na obrázku 1 si pak můžeme prohlédnout výsledek v prohlížeči. (Pevně doufám, že fundamentální příznivci Linuxu přimhouří oko nad obrázkem pořízeným ve Windows, pro jejich uklidnění jsem alespoň použil prohlížeč Mozilla místo Internet Exploreru;–)

Příklad 2: Styl pro konverzi do HTML

```
<?xml version="1.0" encoding="iso-8859-2"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">

<xsl:output method="html"/>

<xsl:template match="prednasky">
  <html>
    <head><title><xsl:value-of select="akce"/></title></head>
    <body>
      <xsl:apply-templates select="akce"/>
      <table border="1">
        <tr>
          <th>Název</th>
          <th>Přednášející</th>
          <th>Firma</th>
          <th>Abstrakt</th>
        </tr>
        <xsl:apply-templates select="prednaska"/>
      </table>
    </body>
  </html>
</xsl:template>
```

```
<xsl:template match="akce">
  <h1 align="center"><xsl:apply-templates/></h1>
  <h2 align="center">Program akce</h2>
</xsl:template>

<xsl:template match="prednaska">
  <tr><xsl:apply-templates/></tr>
</xsl:template>

<xsl:template match="nazev">
  <td><b><xsl:apply-templates/></b></td>
</xsl:template>

<xsl:template match="autor">
  <td><xsl:apply-templates/></td>
</xsl:template>

<xsl:template match="firma">
  <td><xsl:apply-templates/></td>
</xsl:template>

<xsl:template match="abstrakt">
  <td><small><xsl:apply-templates/></small></td>
</xsl:template>

</xsl:stylesheet>
```

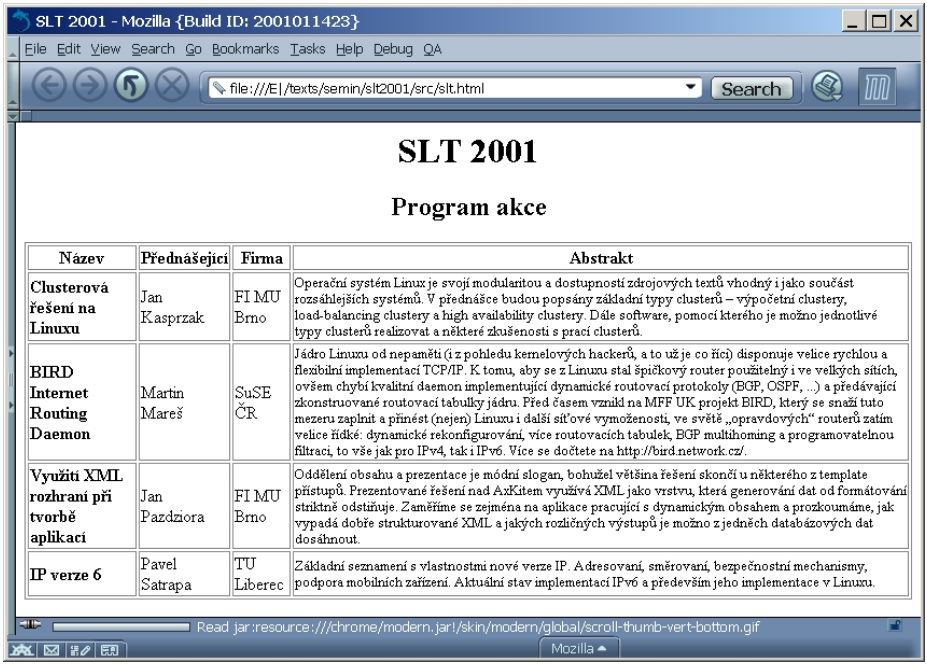
Výhoda tohoto přístupu spočívá v tom, že stylů si můžeme vytvořit několik. Můžeme vytvořit styl, který místo do tabulky uspořádá údaje přehledně pod sebe, nebo například příspěvky nejprve seřadí podle jejich názvu.

Pokud si chceme práci s XSLT vyzkoušet, musíme mít k dispozici nějaký XSLT procesor. Dnes jich existuje několik, mezi asi tři nejznámější open-source procesory patří Saxon, Xalan a XT. Odkazy na ně jsou uvedeny na konci příspěvku. Novější verze Mozilly mají XSLT procesor zabudovaný přímo do sebe.

Formátovací objekty

Formátovací objekty abstraktním způsobem popisují výsledný vzhled dokumentu. Formálně jsou objekty zapsány jako XML dokument, který obsahuje speciální elementy odpovídající jednotlivým formátovacím objektům.

Pokud situaci zjednodušíme, obsahuje soubor s formátovacími objekty nejprve definici layoutu jednotlivých typů stránek použitých v dokumentu – jejich rozměry, velikost okrajů, počet sloupců apod. Za ní jsou už uloženy elementy, které zastupují obsah jednotlivých stránek. XSL procesor pak stránky na výstupu vytváří tak, že do prostoru definovaného layoutem stránky umisťuje obsah těchto elementů.



Obrázek 1: HTML stránka vytvořená z XML pomocí XSLT stylu

Na nejnižší úrovni se pak používají objekty, které zastupují základní druhy formátovaného výstupu. Mezi ty nejpoužívanější patří:

`block` Objekt odpovídá blokovým elementům, které známe z kaskádových stylů. Typicky se používá se pro odstavce, nadpisy apod.

`external-graphic` Objekt zastupuje obrázek, který je uložen mimo výsledný dokument formátovacích objektů. Obvykle je obrázek uložen v externím souboru (např. GIF, JPEG, PNG, EPS apod.).

`float` Plovoucí objekt – umístí se na vhodné místo stránky. Obvykle se používá pro obrázky a tabulky případně pro sazbu poznámek vedle textu (marginálií).

`footnote`, `footnote-citation` Objekty se používají pro poznámky pod čarou.

`inline` Formátovací objekt nezpůsobující vznik nového odstavce. Používá se například pro změny druhu písma uvnitř odstavce.

`leader` Objekt se používá pro čáry nebo opakované znaky (nejčastěji tečky), které mají vyplnit daný prostor. Používá se například v obsahu pro oddělení názvu kapitoly od čísla strany.

`list-block`, `list-item`, `list-item-body`, `list-item-label` Objekty se používají pro seznamy.

`basic-link` Umožňuje do výsledného dokumentu zařadit odkazy.

`table`, `table-*` Několik objektů, které umožňují vytváření tabulek.

`marker`, `retrieve-marker` Objekty umožňují vytváření záhlaví a zápatí, které obsahují proměnlivé texty – např. názvy kapitol a podkapitol.

`page-number`, `pagenumber-citation` Objekty umožňují generování čísla stránky a čísla stránky s určitým objektem.

`wrapper` Objekt se používá v případech, kdy je potřeba pro několik objektů nastavit společné vlastnosti.

U každého z těchto objektů je možné nastavovat několik desítek parametrů, které ovlivní formátování – velikost a typ písma, okraje, různé velikosti odsazení, způsob zarovnání textu, barvy atd. Možnosti formátovacích objektů jsou kompletně popsány ve specifikaci XSL [XSL].

Specifikace XSL se již blíží do finálního stádia, a existuje již i několik jejích implementací. Komerční implementace nabízí firmy RenderX, Unicorn a ArborText, mezi open-source implementace patří FOP a PassiveTEX. Míra podpory formátovacích objektů je v jednotlivých implementacích různá, žádná implementace zatím není úplná. S formátovačem od ArborTextu jsem nepracoval, z těch ostatních je pouze PassiveTEX schopen zpracovat dokumenty obsahující české znaky s diakritikou. Ostatní procesory neobsahují potřebné mapovací tabulky ze znakové sady XML do jednotlivých fontů.

PassiveTEX

Pro první pokusy s formátovacími objekty je PassiveTEX jedním z nejvhodnějších kandidátů. Kromě toho, že je zadarmo a obsahuje poměrně slušnou podporu formátovacích objektů, je schopen sázet české znaky a používat české (a samozřejmě i slovenské) vzory pro dělení slov. Náskok PassiveTEXu oproti ostatním technologiím je možný díky tomu, že PassiveTEX využívá formátovací jádro TEXu a některé možnosti LATEXu.

Jak tedy PassiveTEX funguje? První věc, kterou je potřeba v PassiveTEXu vyřešit, je načítání XML dokumentu s formátovacími objekty. Využívá se xmltex, což je parser napsaný Davidem Carlislem přímo v TEXu. Tento parser umožňuje definovat pro každý element sekvenci TEXových příkazů, které se mají provést na začátku a na konci elementu.

PassiveTEX pak definuje sadu maker, která jsou vystavená nad nízkoúrovňovými příkazy LATEXu a jsou namapovaná na jednotlivé formátovací objekty. Díky PassiveTEXu můžeme TEX použít jako jeden z procesorů pro zpracování formátovacích objektů. Můžeme tak dostat všechny výstupní formáty, které podporuje TEX. V dnešní době je nejatraktivnější výstup do formátu PDF, který je možný díky pdfTEXu.

Malá ukázka použití

Dejme tomu, že náš přehled přednášek budeme chtít zformátovat do tištěné podoby. Vytvoříme proto styl, který převede XML dokument na formátovací objekty. Styl musí obsahovat XSLT instrukce a formátovací objekty. Instrukce řídí zpracování stylu a v našem případě například seřadí přednášky podle názvu.

Příklad 3: Styl pro převod do formátovacích objektů – styl.xsl

```
<?xml version="1.0" encoding="iso-8859-2"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:fo="http://www.w3.org/1999/XSL/Format"
                version="1.0">

<xsl:template match="prednasky">
  <fo:root>
    <fo:layout-master-set>
      <fo:simple-page-master master-name="page"
                             page-height="100mm" page-width="148.5mm"
                             margin-top="10mm" margin-bottom="10mm"
                             margin-left="10mm" margin-right="10mm">
        <fo:region-body
                        margin-top="0mm" margin-bottom="0mm"
                        margin-left="0mm" margin-right="0mm"/>
      </fo:simple-page-master>
    </fo:layout-master-set>
    <fo:page-sequence master-name="page">
      <fo:flow flow-name="xsl-region-body">
        <xsl:apply-templates select="akce"/>
        <xsl:apply-templates select="prednaska">
          <xsl:sort select="nazev"/>
        </xsl:apply-templates>
      </fo:flow>
    </fo:page-sequence>
  </fo:root>
</xsl:template>

<xsl:template match="akce">
  <fo:block text-align="center" font-family="Helvetica"
            font-size="12pt" space-after="4pt">
    <xsl:apply-templates/>
  </fo:block>
  <fo:block text-align="center" font-family="Helvetica"
            font-size="8pt" space-after="5pt">Program akce</fo:block>
</xsl:template>

<xsl:template match="prednaska">
  <fo:block text-align="justify" space-before="6pt"
            font-size="7pt" font-family="Times Roman">
    <fo:block text-align="start" font-weight="bold" font-family="Helvetica">
      <xsl:value-of select="nazev"/>
```

```
    </fo:block>
    <fo:block text-align="start">
      <fo:inline font-weight="bold"><xsl:value-of select="autor"/></fo:inline>
      <xsl:if test="firma">
        <fo:inline font-style="italic">
          (<xsl:value-of select="firma"/>)</fo:inline>
      </xsl:if>
    </fo:block>
    <xsl:apply-templates select="abstrakt"/>
  </fo:block>
</xsl:template>

<xsl:template match="abstrakt">
  <fo:block font-size="6pt" space-before="1pt">
    <xsl:apply-templates/>
  </fo:block>
</xsl:template>

</xsl:stylesheet>
```

Nejprve musíme pomocí tohoto stylu převést vstupní XML dokument `slt.xml` na XML dokument s formátovacími objekty. K tomu potřebujeme nějaký XSLT procesor. Například s procesorem XT [XTCS] můžeme transformaci provést následujícím příkazem:

```
xt slt.xml styl.xsl slt.fo
```

Dostaneme dokument s formátovacími objekty (viz ukázka 4), který můžeme zformátovat libovolným procesorem formátovacích objektů. V našem případě použijeme PassiveTEX. Pro jeho spuštění je potřeba nějaká novější distribuce TEXu, například TEXLive 5. Ten obsahuje i xmltex, je však dobré stáhnout si vždy nejnovější verzi PassiveTEXu [PTEX], protože se neustále rychle vyvíjí. K zformátování do PDF (výsledek je na obrázku 2) pak použijeme příkaz:

```
pdfxmltex slt.fo
```

Příklad 4: Výsledný dokument s formátovacími objekty

```
<?xml version="1.0" encoding="utf-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <fo:layout-master-set>
    <fo:simple-page-master master-name="page" page-height="100mm"
                           page-width="148.5mm" margin-top="10mm"
                           margin-bottom="10mm" margin-left="10mm"
                           margin-right="10mm">
      <fo:region-body margin-top="0mm" margin-bottom="0mm"
                           margin-left="0mm" margin-right="0mm"/>
    </fo:simple-page-master>
  </fo:layout-master-set>
  <fo:page-sequence master-name="page">
    <fo:flow flow-name="xsl-region-body">
```

```
      <fo:block text-align="center" font-family="Helvetica"
                font-size="12pt" space-after="4pt">SLT 2001</fo:block>
      <fo:block text-align="center" font-family="Helvetica"
                font-size="8pt" space-after="5pt">
        Program akce
      </fo:block>
      <fo:block text-align="justify" space-before="6pt"
                font-size="7pt" font-family="Times Roman">
        <fo:block text-align="start" font-weight="bold"
                  font-family="Helvetica">BIRD Internet Routing Daemon</fo:block>
        <fo:block text-align="start">
          <fo:inline font-weight="bold">Martin Mareš</fo:inline>
          <fo:inline font-style="italic">(SuSE ČR)</fo:inline>
        </fo:block>
        <fo:block font-size="6pt" space-before="1pt">Jádro Linuxu od
          nepaměti (i z pohledu kernelových hackerů, a to už je co
          říci) disponuje velice rychlou a flexibilní implementací
          ...
          jak pro IPv4, tak i IPv6. Více se dočtete na
          http://bird.network.cz/.</fo:block>
      </fo:block>

      ...

    </fo:flow>
  </fo:page-sequence>
</fo:root>
```

Další možnosti

PassiveTEX obsahuje dvě velice užitečná rozšíření. Prvním z nich je možnost generovat automaticky záložky (bookmarks) v PDF dokumentu. Kromě formátovacích objektů XSL, rozeznává PassiveTEX další element, který umožňuje specifikovat název záložky, její úroveň a místo v dokumentu, kam má ukazovat.

Druhé rozšíření dovoluje využívat nejsilnější stránku TEXu – sazbu matematických vzorců. Existuje speciální jazyk MathML (Mathematical Markup Language), který umožňuje zapisovat matematické výrazy v syntaxi založené na XML. Této syntaxi dnes rozumí několik editorů vzorců, prohlížeče Mozilla a Amaya a i některé matematické programy. PassiveTEX umí podmnožinu MathML přímo formátovat, takže pokud chceme mít matematické vzorce na Webu i na papíře, je použití MathML a PassiveTEXu jednou z možností.

Závěr

Ačkoliv PassiveTEX zatím neimplementuje celý návrh XSL, na mnoho věcí již stačí a dá se bez problémů použít. Jeho velkou výhodou je využití kvalitních

BIRD Internet Routing Daemon
Martin Mareš *(SuSE ČR)*
Jádro Linuxu od nepaměti (i z pohledu kernelových hackerů, a to už je co říci) disponuje velice rychlou a flexibilní implementací TCP/IP. K tomu, aby se z Linuxu stal špičkový router použitelný i ve velkých sítích, ovšem chybí kvalitní daemon implementující dynamické routovací protokoly (BGP, OSPF, ...) a předávající zkonstruované routovací tabulky jádru. Před časem vznikl na MFF UK projekt BIRD, který se snaží tuto mezeru zaplnit a přinést (nejen) Linuxu i další síťové vymoženosti, ve světě „opravdových" routerů zatím velice řídké: dynamické rekonfigurování, více routovacích tabulek, BGP multihoming a programovatelnou filtraci, to vše jak pro IPv4, tak i IPv6. Více se dočtete na http://bird.network.cz/.

Clusterová řešení na Linuxu
Jan Kasprzak *(FI MU Brno)*
Operační systém Linux je svojí modularitou a dostupností zdrojových textů vhodný i jako součást rozsáhlejších systémů. V přednášce budou popsány základní typy clusterů – výpočetní clustery, load-balancing clustery a high availability clustery. Dále software, pomocí kterého je možno jednotlivé typy clusterů realizovat a některé zkušenosti s prací clusterů.

IP verze 6
Pavel Satrapa *(TU Liberec)*
Základní seznamení s vlastnostmi nové verze IP. Adresovaní, směrovaní, bezpečnostní mechanismy, podpora mobilních zařízení. Aktuální stav implementací IPv6 a především jeho implementace v Linuxu.

Využití XML rozhraní při tvorbě aplikací
Jan Pazdziora *(FI MU Brno)*
Oddělení obsahu a prezentace je módní slogan, bohužel většina řešení skončí u některého z template přístupů. Prezentované řešení nad AxKitem využívá XML jako vrstvu, která generování dat od formátování striktně odstiňuje. Zaměříme se zejména na aplikace pracující s dynamickým obsahem a prozkoumáme, jak vypadá dobře strukturované XML a jakých rozličných výstupů je možno z jedněch databázových dat dosáhnout.

Obrázek 2: Zformátovaný dokument

typografických algoritmů. Uživatel tak může využít typografický potenciál TEXu bez znalosti jeho makrojazyka, který, přiznejme si, není úplně pro každého. Využití XML a XSLT přináší navíc další výhody – úplné oddělení obsahu od vzhledu a pohodlnější transformace vstupních dat před sazbou.

Odkazy

[MATHML] Mathematical Markup Language (MathML) Version 2.0 [`http://www.w3.org/TR/MathML2`]. W3C.

[PTEX] Rahtz, Sebastian: PassiveTeX. `http://users.ox.ac.uk/~rahtz/passivetex/`

[SAXON] Kay, Michael H.: XSLT procesor Saxon. `http://users.iclway.co.uk/mhkay/saxon/`

[XALAN] XSLT procesor Xalan. `http://xml.apache.org/xalan/`

[XSL] Adler Sharon, Berglund Anders, Caruso Jeff, Deach Stephen, Grosso Paul, Gutentag Eduardo, Milowski Alex, Pernell Scott, Richman Jeremy, Zilles Steve: Extensible Stylesheet Language (XSL) – Version 1.0 [`http://www.w3.org/TR/xsl`]. W3C, 2000.

[XSLT] Clark, James: XSL Transformations (XSLT) Version 1.0 [`http://www.w3.org/TR/xslt`]. W3C, 1999.

[XT] Clark, James: XSLT procesor XT. http://www.jclark.com/xml/xt.html
[XTCS] Kosek, Jiří: XT s podporou českých kódování. http://www.kosek.cz/xml/xt/
[ZVON] Tutoriály XSLT a XSL na Zvonu. http://www.zvon.org

Summary: PassiveTEX

PassiveTEX is a macropackage suited for formatting XML documents. Layout of the output document is described by an XSL stylesheet. Creating XSL stylesheets is easier than writing TEX macros for most users. PassiveTEX hides TEX internals to a common user, but a user can still easily use the state of the art typographic features of the TEX system. This article introduces XSL and PassiveTEX usage on a few easy examples.

Nový vizuální styl CSTUGu

Zdeněk Wagner

Začátkem roku 2000 přijal výbor Československého sdružení uživatelů TEXu nové logo. Autorem loga je Antonín Strejc. Logo je složeno z písmen z fontu New Century Schoolbook a dvou linek. Linky a střední písmeno T jsou červené. Písmena symbolizují anglickou zkratku „Czechoslovak TEX Users Group“ (českou zkratku CSTUG nemá) a jsou graficky uspořádána do podoby map ČR a SR.

Černobílou podobu loga jste mohli poprvé spatřit ve Zpravodaji č. 4/1999 na straně 235. Barevná verze byla použita na nových vydáních knih „Typografický systém TEX“ a „TEXbook naruby“.

S novým logem souvisí též nový vizuální styl všech tiskovin. Základní návrh také vytvořil Antonín Strejc. Nejvýznamnější je vzhled hlavičkového dopisního papíru. Všichni členové dostali od výboru dopis v lednu roku 2001. Tento dopis již využívá nový vizuální styl.

Do jubilejního 10. ročníku vkročil i Zpravodaj s novou obálkou, navrženou Antonínem Strejcem. Nová obálka z důvodu zachování kontinuity vychází z grafického vzhledu předchozích ročníků. Nový design nejen že z obálky odstraňuje matoucí neoficiální název, ale hlavně přináší nové logo a uvolňuje dostatek prostoru pro umístění čtyř číslic roku, což řeší problém Y2K.

Ve spolupráci s Evou Žáčkovou bylo nové logo zakomponováno též do WWW stránek sdružení. Stránky v této podobě se dostaly k široké veřejnosti, neboť jejich statická kopie byla zveřejněna na CD v únorovém čísle časopisu Chip.

Nová je též stránka `https://www.cstug.cz/cstug/admin/clen.php3`, kde má každý člen přístup ke svým vlastním informacím. V minulosti docházelo k problémům, že nebyl doručen Zpravodaj, protože naše údaje neodpovídaly skutečnosti. Nyní si každý člen své údaje může zkontrolovat a opravit.

Nový vzhled WWW stránek, jež se objevily na Internetu několik dní před zahájením SLT, představuje tak první vykročení CSTUGu do nového tisíciletí.

Grantový systém CSTUGu

Zdeněk Wagner

CSTUG má ve svých stanovách zakotvenu podporu softwarového vybavení pro kvalitní českou a slovenskou typografii, zejména s ohledem na TEX a související programy. V souladu s tímto článkem stanov se výbor rozhodl, že založí grantový systém, který umožní soutěžit o projekty pro tvorbu nových a vylepšování stávajících programů, o reprezentaci na mezinárodních akcích, a podobně. Stanovy tohoto grantového systému, za jehož chod odpovídají Zdeněk Wagner a Jiří Veselý, najdete na následujících stránkách Zpravodaje i na WWW stránce `http://bulletin.cstug.cz/igscstug/`.

Zvláštní kategorii tvoří tzv. zamýšlené projekty. Jejich účelem je určit, jaký software je pro české a slovenské uživatele potřebný a prioritní. Zamýšlený projekt má konkrétního navrhovatele, který specifikuje požadavky na software, který má být vyvinut nebo upraven. Řešitel je pak určen z výsledků grantové soutěže.

Veškerá agenda bude zajišťována výhradně prostřednictvím WWW a e-mailu (`gacstug@cstug.cz`). Rozcestníkem je výše zmíněná stránka. Na ní najdete odkaz na stanovy grantového systému a na seznam schválených zamýšlených projektů. Zadání každého zamýšleného projektu obsahuje tlačítko, po jehož stisknutí se objeví formulář, v němž je nutno vyplnit předepsané údaje.

Vzhledem k tomu, že grantový systém letos poprvé vstupuje do života, jsou některé termíny výjimečně posunuty.

Zdeněk Wagner
`gacstug@cstug.cz`

Stanovy interního grantového systému Československého sdružení uživatelů TEXu

Interní grantový systém zřizuje CSTUG s cílem podporovat kvalitní typografii, zejména pak prostřednictvím zdokonalování a vývoje TeXu a jeho podpůrných programů v ČR a SR.

1.

Z grantového fondu se udělují finanční podpory pro realizaci projektů, které výbor CSTUGu shledá na základě odborného posouzení dvěma nezávislými oponenty jako celkově nejlepší. Doba realizace grantu nesmí přesáhnout jeden rok. Ve výjimečných případech ji lze prodloužit opakováním návrhu doprovozeného přehledem dosažených výsledků. Celková výše finančních prostředků přidělených na realizaci projektů je stanovena vždy pro jednoleté období na návrh výboru CSTUGu členským shromážděním. Rovněž výše prostředků přidělených projektu nelze v průběhu jeho realizace měnit. Každý grant musí mít navrhovatele a v případě realizace i řešitele.

2.

Navrhovatelem může být fyzická i právnická osoba, výbor apod. O udělení grantu se ucházejí fyzické osoby; v případě, že nejsou členy CSTUGu musí být žádost o podporu projektu doporučeno dvěma řádnými členy CSTUGu. Závazný termín pro podávání návrhů na řešení projektů je vždy 1. říjen roku předcházejícímu rok realizace projektu. Grant musí být ukončen a oponován do 30. listopadu dvěma nezávislými oponenty. Podle povahy projektu je podpora poskytována předem, výbor však může vázat výplatu části finančních prostředků danému projektu na úspěšné ukončení projektu. V případě výrazně rozdílného posouzení oběma posuzovateli je možno požádat o třetí nezávislý posudek.

3.

Samotný návrh grantu je rozsahem omezen na jednu normostránku (30 řádek). Vyžaduje-li to povaha projektu, doplní navrhovatel svůj návrh přílohami, které v písemné podobě doručí na adresu výboru CSTUGu do 14 dnů po zveřejnění návrhu na síti, ne však později než do 30. října. Výbor si může v odůvodněných případech vyžádat podrobnější návrh grantu. Navrhovatel je pak povinen

dodat podrobnější návrh písemně na adresu CSTUGu do 14 dnů od obdržení výzvy. Výbor rozhoduje o zveřejnění grantu elektronickým hlasováním, o udělení podpory a její výši na své schůzi zpravidla předcházející výročnímu shromáždění v prosinci.

4.

Navrhovatel může být zároveň i řešitelem grantu, mohou však jimi být i rozdílné subjekty. Návrh může být podán s cílem přispět k řešení aktuálního problému jiným subjektem, např. včetně navrhované odměny. Podpora se však poskytuje řešiteli na základě jednání o jeho návrhu výše grantu, který může být odlišný. V tomto případě si vyžádá výbor kromě obou nezávislých posudku i posudek navrhovatele.

5.

Závěrečná zpráva řešitele grantu musí jasným způsobem popsat využití finančních prostředků a toto využití průkazným způsobem doložit účetními doklady. Obsah zprávy může být při rozhodnutí o udělení podpory výborem na základě dohody s řešitelem blíže specifikován.

6.

Projekt navržený k řešení bez určení řešitele (dále zamýšlený projekt) je třeba předložit vždy do 1. března roku, předcházejícímu kalendářnímu roku, pro který by měly být projektu alokovány prostředky. Projekt je automaticky zveřejněn, podpoří-li ho nadpoloviční většina výboru CSTUGu v elektronickém hlasování. Případné úpravy musí být dohodnuty s navrhovatelem zamýšleného projektu. Při jeho nesouhlasu musí výbor zveřejnit projekt pod svým jménem s výslovnou poznámkou o autorovi původního záměru. Náležitosti podání zamýšleného projektu vymezuje přesněji jeho elektronický formulář, dostupný na WWW stránkách CSTUGu. Obsahuje mj. autora zamýšleného projektu, textovou část projektu, návrh na výši alokovaných prostředků, návrh kontroly výsledků projektu.

7.

Návrh projektu předkládaný řešitelem na zamýšlený projekt obsahuje rozšiřující údaje, které doplňují zamýšlený projekt. Část zamýšleného projektu se tak stává automaticky součástí předkládaného projektu. Projekt, který nenavazuje na žádný zamýšlený projekt, obsahuje zejména tyto náležitosti:

Jméno a příjmení řešitele,
název projektu,
termín dokončení,
pracoviště,
adresa, (bankovní spojení, účet),
text projektu,
návrh na přidělení částky,
charakteristiku a kvalifikaci řešitele
(vzdělání, praxe, realizované výsledky práce vztahující se k projektu, atp.)
rozpis nákladů

Rozpis celkových nákladů, který je pak při realizaci nutno dodržet, je třeba předložit v každém případě, stejně tak jako návrh na každou jeho změnu. Ta je možná, avšak pouze po schválení výborem ČSTUGu.

8.

O oponentech rozhodne výbor ČSTUGu elektronickým hlasováním na základě dobrozdání svého určeného člena. Toho navrhuje předseda ČSTUGu a schvaluje výbor (nutná nadpoloviční většina kladných hlasů). Oponenti zpracují posudky, které mohou být honorovány, ve vymezené lhůtě Projekt se posuzuje neanonymně, s posudky je seznámen výbor. Text posudků si může vyžádat navrhoval, není mu však sdělena totožnost posuzovatelů. Výbor rozhodne po seznámení s posudky o pořadí přijatých projektů a o alokovaných částkách. Pokud nedojde ke snížení, je tím projekt přijat, v opačném případě může řešitel od řešení do 14 dnů odstoupit. Pořadí přijatých projektů je rozhodující pouze v tom smyslu, že při odmítnutí realizace řešitelem je vyzván další řešitel dle pořadí, pokud ovšem je možná alokovaná částka dostačující.

9.

Autorská práva na software vytvořený v rámci grantového projektu má řešitel. Projekt musí vyhovovat GNU GPL, pro knihovny GNU LGPL, přičemž pro Českou republiku se použijí smlouvy ve znění, které připravilo občanské sdružení *zastudena.cz*. ČSTUG se stane distributorem SW a držitelem práva na užívání tohoto SW zdarma. V odůvodněných případech lze dohodnout jiné podmínky, např. bezplatné používání pouze pro členy ČSTUGu. Tyto podmínky však musí být písemně zachyceny formou dohody mezi řešitelem a ČSTUGem, která musí být uzavřena před zahájením práce na řešení projektu.

10 .

Ve sporných případech upřesnění a případně výklad ustanovení těchto stanov provádí výbor CSTUGu.

Zamýšlený projekt – úprava programu csbibtex

Cíl:

Upravit stávající program csbibtex autora Petra Novotného, který je určen především do instalace emtexu, do instalací postavených na web2c. Autor poskytl svoje úpravy programu bibtex, jehož zdrojový kód je volně k dispozici a je součástí standardních instalací web2c, a přislíbil poskytnout případné další informace řešiteli. (Úpravy, které provedl Eberhard Mattes, nemohou být volně distribuovány.)

Požadavky:

1) Správné třídění podle pravidel češtiny a slovenštiny (na přepínač).
2) Zvládnutí i velkých databází (vstup z více souborů).
3) Použití nejnovějšího balíku web2c (momentálně ver. 7.3.2).
4) Vyhledávání vstupních souborů pomocí knihovny kpathsea.
5) Akceptovat osmibitový vstup.
6) Vstupní a výstupní kódování řešit pomocí tcx tabulek místo alp tabulek použitých v csbibtexu.
7) Umět přepínače /ch, /noch a /and jako stávající verze csbibtexu.
8) Mít přepínač, aby se choval jako normální bibtex (sedmibitový nebo osmibitový).
9) Nápověda česky, slovensky a anglicky (na přepínač).
10) Licence GPL.

Výstup:

1) Zdrojový kód (asi doplňky ke stávajícímu bibtexu) a makefile, které bude možné přeložit pro běžné instalace postavené na web2c, které jsou integrovatelné do balíku TEXlive 6.
2) Binární soubor pro linux a win32.
3) Dokumentace (uživatelská, ke zdrojovému kódu).
4) Závěrečná zpráva a publikace do Zpravodaje $\mathcal{CS}$TUGu.

Navrhovaná odměna:

5 000–10 000 Kč podle úrovně řešení.

Uzávěrka přihlášek:

20. 6. 2001

Termín:

Do 6 měsíců od přidělení grantu.

Přihláška:

Přihlášku je nutno podat elektronicky formulářem, který je dostupný ze stránky `http://bulletin.cstug.cz/igscstug`, v případě nejasností se můžete obrátit e-mailem na `gacstug@cstug.cz`.

Call for Contributions for EuroTEX 2001

The Program Committee

The EuroTEX conference 2001 will take place from September 23rd to September 27th in the Netherlands. The conference will be held in an old abbey, Rolduc, near the city of Kerkrade.

Participants will be welcomed on Sunday evening the 23rd of September. The conference proper will start on Monday and continue until Wednesday. On Thursday courses and tutorials may be given.

The theme of the conference is:

TEX and Meta
the Good, the Bad and the Ugly Bits

So, what are "Good, Bad and Ugly Bits"? Traditionally we tend to focus on Good Bits because they are so rewarding. It feels good to see a beautifully typeset document that you were able to produce using all the power that TEX and METAFONT/METAPOST provide. We try to forget how many Bad Bits we had to overcome, and are proud that we found more or less elegant ways to program around them. Nevertheless, some of these difficulties should be regarded as Ugly Bits that would not exist in an ideal world.

What can we do?

First of all, keep up the Good Bits and extend them if possible. Analyze the Bad Bits, learn from them, and find easy and generic ways to get around them. Find the Ugly Bits and eradicate them!

Let's make progress! Of course we need a lot of discussion to distinguish between Good, Bad and Ugly, and about ways to deal with them. We invite you all to demonstrate either of these bits. Feel free to present your application (Good, not Bad or Ugly) at the conference!

At this conference we would like to take a new view on contributions.

- Of course we welcome a presentation accompanied by a written paper, as usual.
- But does it bother you to write a paper, or does your contribution seem fit for an interesting presentation but not for a written paper? We will welcome the presentation equally much, and do with only a short summary on paper or even entirely without the paper work.
- Or do you find yourself in the opposite situation: You would like to clarify your new package or other work extensively on paper but believe that it would be unsuitable to cover it equally extensively in an oral presentation? Feel free to let us have your paper and at the conference present only the highlights.
- Or do you believe your ideas only warrant a short presentation without a paper? It is equally welcome.

We do expect from each contributor a *short summary* of the proposed contribution. This summary will inform the Program Committee of your plan. And at the conference it will be the introduction to your contribution, or your complete contribution if that is what you propose.

We invite you to submit your proposal for a contribution to the conference. You can send it by email to `eurotex-pc@ntg.nl`. If you have no access to e-mail

you can send your contribution to:
EuroTEX 2001 program committee
Spuiboulevard 269
3311 GP Dordrecht
The Netherlands.
In such a case we would ask you to submit your paper in electronic form on a floppy disk.

We are working with the following time schedule:

May 10	deadline for receiving proposals
June 10	deadline for receiving full draft papers
July 11	end of review
August 11	deadline for receiving final, reviewed papers

An up-to-date list of contributions as well as more information about the conference can be found at `www.ntg.nl/eurotex`.

9th GUST Annual Meeting, April 29 – May 1, 2001, Bachotek (Brodnica Lake District, Poland)

The 9th annual meeting of the Polish TeX Users' Group (GUST) will be held in Bachotek, Brodnica Lake District, April 29 – May 1, 2001. We invite TeX users, no matter from where, to join us. The topic of the meeting is

Contemporary Publishing TEXnology

During this year's conference we will concentrate on the practical aspects of how to produce scientific and technical documents using TEX. Papers on related topics are welcomed too.

The list of possible topics includes:

- Tools: TEX-ware, editors, drivers etc.;
- Specialized macros and formats;
- Fonts;
- Multimedia publications;
- PostScript, PDF, SGML, HTML, XML, MathML, tools and applications;
- Paper publications vs electronic publishing;
- Textual databases;
- Typography issues;

- Standards;
- Other.

As always, the conference programme will be completed with TUTORIALS. We expect to have the following tutorials (all in Polish):

- Graphics in TEX and pdfTEX (Piotr Bolek)
- XSL – a language to process XML documents (Mariusz Olko)
- Postscript not only for printers: stacks, dictionaries and filters (Piotr Strzelczyk)
- LATEX3: evolution or revolution (Marcin Wolinski)

A preliminary programme inludes: macros to draw graphs, macros to count in TEX, Type3 fonts, multilingual texts in GNU emacs, input and output in (x)emacs, index processor xindy and the New Typesetting System.

Paper submissions and remarks relating to the programme of the meeting should be sent to the Programme Committee: Andrzej Borzyszkowski, e-mail: `bachotek@gust.org.pl`

Lectures might also be held in English. The deadline for submissions is March 30, 2001.

It is planned that the conference proceedings will be published as a special issue of the GUST bulletin. The programme committee strongly encourages authors to provide written submissions to accompany oral presentations. The deadline for written submissions is April 6, 2001.

The conference fee (without accommodation or catering) is 440 PLN. A special fee for LUG memebers is 340 PLN (1 PLN is approx 0,24 US dollar). Student fees are 250 PLN and 200 PLN, respectively.

Accomodation and catering cost is 73,20 PLN per day.

The delegates are expected on April 28th (Saturday, afternoon). It is possible to extend the stay in Bachotek to May 6th (Sunday) for additional 73,20 PLN per day.

For more information, please contact the GUST secretary, Jola Szelatynska, Uniwersyteckie Centrum Komputeryzacji UMK, Gagarina 7, 87-100 Torun, POLAND (`secretary@gust.org.pl`).

`http://www.gust.org.pl/BachoTeX/2001-e.html`

Please send the registration form before 18th April, 2001 to the organizers: Jola Szelatynska, Uniwersyteckie Centrum Komputeryzacji UMK, Gagarina 7, 87-100 Torun, POLAND (or via email: `secretary@gust.org.pl`).

The payment should be made to: Bank Gdanski, I Oddzial w Toruniu, Polska Grupa Uzytkownikow Systemu TeX GUST, account number: 11601612-5890-132, or in cash on arrival.

Zpravodaj Československého sdružení uživatelů TEXu
ISSN 1211-6661

Vydalo:	Československé sdružení uživatelů TEXu vlastním nákladem jako interní publikaci
Obálka:	Antonín Strejc Na obálce použito logo SLT, jehož autorkou je Petra Rychlá
Počet výtisků:	750
Uzávěrka:	22. února 2001
Odpovědný redaktor:	Zdeněk Wagner
Tisk a distribuce:	KONVOJ, spol. s r. o., Berkova 22, 612 00 Brno, tel. 05-740233
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

gacstug@cstug.cz
grantová agentura CSTUGu

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

cstug-members@cstug.cz
korespondence členům sdružení

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

bookorders@cstug.cz
objednávky tištěné TEXové literatury na dobírku

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

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

Podávání novinových zásilek povoleno Českou poštou, s. p. OZJM Ředitelství v Brně č. j. P/2–1183/97 ze dne 11. 3. 1997.