Zpravodaj Československého sdružení uživatelů TeXu

Michal Kvasnička
ConTeXt

Zpravodaj Československého sdružení uživatelů TeXu, Vol. 10 (2000), No. 4, 93–112

Persistent URL: http://dml.cz/dmlcz/150245

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.