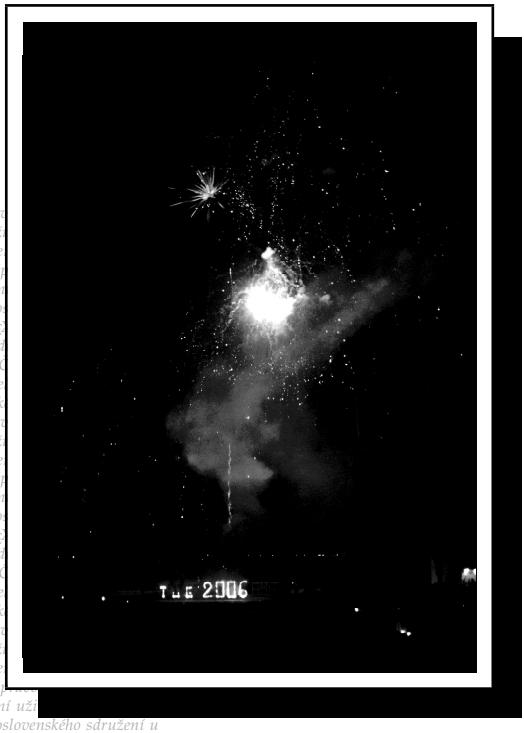
CSTUG



ZPRAVODAJ

Československého sdružení uživatelů TEXu

1

2007

ISSN 1211-6661 Tištěná verze

ISSN 1213-8185 Online verze

Ročník 17

OBSAH

Petr Tesařík: S češtinou a slovenštinou do Babylónu 2

Zdeněk Wagner: Babylón mluví hindsky 12

Zdeněk Wagner: Babylón v TEX Live 2007 21

Petr Březina: Abecední řazení a sestavování rejstříků výhradně pomocí makrojazyka TEXu 23

Pavel Stříž: Proměnné záhlaví a zápatí 31

Zdeněk Wagner: 27th TUG Annual Meeting, Marrákéš 2006, krátká reportáž 60

Součástí tototo čísla je DVD TEX Collection 2007.

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

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

Zdeněk Wagner
Vinohradská 114
130 00 Praha 3

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

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

Milí přátelé.

Do rukou vám přichází první číslo Zpravodaje CSTUGu roku 2007. Ti z vás, kteří byli členy CSTUGu již v loňském roce a zaplatili příspěvky, v něm naleznou jedno DVD (kolektivní členové tři DVD), obsahující mimo jiné distribuce TEXLive a proTEXt. V případě jakýchkoli nesrovnalostí se prosím obraťte na sekretářku našeho sdružení.

Stala se jí od letošního února paní Dana Čurdová a převzala štafetu po paní Heleně Holovské. Té bych chtěl co nejsrdečněji poděkovat za to, že obětavě a mimořádně zodpovědně vykonávala po pět let tuto nevděčnou práci, a popřát jí mnoho zdraví a osobních úspěchů v dalším životě.

Loni 2. 12. proběhla na FI MU v Brně valná hromada CSTUGu. Doprovodnou akcí byla přednáška pana Petra Tesaříka *Český a slovenský Babylón*. Cílem přednášky bylo seznámit přítomné s novým českým a slovenským definičním souborem pro Babel. Tyto soubory vytvořil p. Tesařík v rámci grantu, který na rok 2006 vypsal CSTUG. Další podrobnosti lze nalézt v tomto čísle Zpravodaje v článku autora.

Bohužel opět je nutné konstatovat, že účast na valné hromadě byla malá. Ta proběhla podle následujícího programu:

1. Zahájení valné hromady.
2. Zpráva o činnosti (J. Kuben).
3. Zpráva o hospodaření (J. Kuben).
4. Revize hospodaření (revizoři P. Sekanina a T. Hála).
5. Informace o přípravě DVD TEXLive 2006 (P. Sojka, Z. Wagner).
6. Rámcový plán činnosti pro rok 2007.
7. Různé, diskuse.
8. Závěr.

V průběhu valné hromady informoval p. Wagner o obsahu zbývajících čísel Zpravodaje 2006. Bylo rozhodnuto, že se bude jednat o trojčíslo s překladem manuálu ConTEXtu, který byl připraven opět v rámci grantu CSTUGu. Dále p. Sojka referoval o přípravě konference EuroBachoTEX. Byla doporučeno podpořit aktivní účast členů CSTUGu.

Potěšitelné je, že se osvědčilo vypsání grantů CSTUGu. Dva projekty jsou hotové (babelizovaný LATEX a překlad manuálu ConTEXtu), třetí (Užitečné Latin Modern pro Čechy, Moraváky a Slováky) je před dokončením. CSTUG by rád vypsal další granty. Výbor má několik tipů, přesto vítá jakékoli vhodné návrhy.

Jednotlivé dokumenty z valné hromady i přednášky lze nalézt na stránkách sdružení `www.cstug.cz`.

Během prvního čtvrtletí tohoto roku výbor schválil podporu projektu TEX Gyre (tvorba kvalitních fontů pro TEX — viz `www.gust.org.pl/e-foundry/ /tex-gyre`). Tím se CSTUG připojil k dalším národním skupinám, které tento

projekt podpořily (Dante, NTG, TUG, TUG India). Podobně výbor rozhodl v budoucnu podpořit projekt mplib (Metapost Library), který by měl vyústit do tvorby programu METAPOST nové generace.

Poslední informace se týká konference EurobachoTEX, na jejíž organizaci se CSTUG podílí. Vzhledem ke skutečnosti, že zájem o autobus, který by „posbíral“ účastníky z České republiky, Slovenska, Maďarska, Rakouska a Německa, byl mizivý, muselo se od jeho organizování upustit. Z členů CSTUGu se aktivně zúčastní pánové K. Horák, P. Sojka a K. Píška. Výbor proto rozhodl, že jim CSTUG částečně přispěje na pokrytí nákladů na účast na konferenci.

Na závěr bych vám všem chtěl popřát v letošním roce, kdy nás čeká mimo jiné volba nového výboru CSTUGu, hodně zdraví a spoustu osobních i TEXovských úspěchů.

Jaromír Kuben
předseda CSTUGu

S češtinou a slovenštinou do Babylónu

Petr Tesařík

Tento článek je jedním z výstupů grantu CSTUGu na vytvoření české a slovenské podpory pro balík Babel, jehož jsem byl řešitelem. Možná se někdo podiví, jak to myslím – vždyť podpora češtiny i slovenštiny je v Babelu již řadu let! A má pravdu: Bylo by vlastně přesnější mluvit o přesunu CSLATEXu pod křídla Babelu.

Něco je špatně

S trochou nadsázky se dá říci, že na počátku veškerého zmatku stál LATEXový amerikanocentrismus či spíše snaha si jej nevšímat. Přestože totiž LATEX nebyl původně určen pro sazbu jiných než anglických textů, objevily se v průběhu let rozličné snahy, jak jej přizpůsobit pro sazbu dokumentů v jiných jazycích, případně i ve více jazycích současně. Pro češtinu a slovenštinu upravili LATEX Jiří Zlatuška a Zdeněk Wagner; výsledný produkt dostal jméno CSLATEX a o něco později jej dále vylepšil Jaroslav Šnajdr. Paralelně se již od počátku devadesátých let vyvíjel balík Babel, který se snažil sjednotit či alespoň zastřešit lokalizaci LATEXu pro nejrůznější jazyky.

Soudě podle názvů některých maker se domnívám, že CSLATEX se snažil být s Babelem spíše kompatibilní. Původní záměr byl možná dokonce takový, aby

styly `czech.sty` a `slovak.sty` byly do Babelu začleněny, jako se to již předtím stalo například se styly německým a francouzským. Nicméně vývoj se ubíral jinou cestou, a tak dnes existují dva soubory `czech.sty` a dva soubory `slovak.sty`, které jsou navzájem nekompatibilní.

Největší závadou je však samotná existence dvou stejně pojmenovaných souborů. Jedním z nejčastějších problémů je tak správné nastavení prohledávaných cest pro babelizovaný LaTeX a pro CSLaTeX. Komu nepřipadá absurdní, že pro správné fungování stylu `czech.sty` je občas nutné nejdříve smazat soubor `czech.sty`, nejspíš výsledky mé práce neocení. Ti ostatní si mohou přečíst, jak nový styl funguje a proč právě tak.

Ach, ty uvozovky...

...aneb proč nelze vytvořit dokonalé makro `\uv`. Jak asi všichni víte, existují v CSLaTeXu dvě verze tohoto makra, jejichž rozdíl bývá vysvětlován následovně:

- „staré“, které umožňuje umístit do uvozovek `\verb`, ale nezachovává implicitní kerny a
- „nové“, které sice neumožňuje použití `\verb`, zato zachovává implicitní kerny na obou stranách.

Skutečnost je ještě o něco barvitější, neboť v nějaké verzi CSLaTeXu byla definice `\crqq` změněna následovně:

```
\DeclareTextCommand{\crqq}{IL2}%
  {{\edef\@SF{\spacefactor\the\spacefactor}\char255 \@SF\relax}}%
```

Nevím, zda je v české či slovenské typografii zvykem, aby se velikost mezery za uvozovkami řídila posledním znakem před uvozovkami, nicméně uvedený kód má za následek, že se mezi tento poslední znak a uvozovky dostane povel hlavního procesoru, takže TeX pak tuto dvojici nehledá v tabulce kerningových párů. Případný implicitní kern u pravé uvozovky se tedy neuplatní (schválně si zkuste jak se Vám s CS-fonty vysází kód `\uv{teď}`). Ironií osudu je, že by bylo celkem snadno možné vytvořit takové makro `\uv`, které by zachovávalo kerny pouze u levé uvozovky a zároveň umožňovalo použít `\verb`...

Rozumně složité makro, které by umožňovalo použít `\verb` a zároveň zachovávalo implicitní kerny na *obou* stranách, vytvořit nelze. Zkusím to jednou pro vždy dokázat. Pro začátek si připomeňme, jak TeX zachází s kerningovými páry (TBN, str. 103):

> Každá dvojice znaků ze společného fontu může mít v tabulce kerningových párů fontu nějakou hodnotu, která se mezi znaky vloží v podobě implicitního kernu automaticky. [...] Pokud se mezi vložením dvojice znaků do horizontálního seznamu vykoná jakýkoli jiný povel hlavního procesoru, implicitní kern se nevloží a ligatura se nevytvoří.

Očividně tedy nelze otevřít skupinu a uvozovky vysázet pomocí `\aftergroup` (trik použitý ve „staré“ definici makra `\uv`), protože mezi vysázením posledního znaku a zpracováním povelů z `\aftergroup` musí hlavní procesor vykonat ukončení skupiny.

Protože `\verb` nemůže být použito v parametru makra (metoda zvolená v „nové“ definici), zbývají jenom následující možnosti:

1. předefinovat kategorii znaku `}`, nebo
2. zpracovat celý obsah uvozovek pomocí `\let`, případně `\futurelet`.

V prvním případě narazíme na to, že je třeba aktivovat také znak `{`. Pokud je totiž uvnitř uvozovek vložena skupina, musí odpovídající znak `}` tuto skupinu ukončit, ovšem bez uvozovek. Problém nastává s tím, že znak `{` se používá také jako levé ohraničení parametru makra. Pokud je tento znak aktivní, použije se jako parametr pouze tento znak. Jakmile je parametr jednou přiřazen, není možné ho změnit, takže tímto směrem se dále nedostaneme.

V druhém případě narazíme na to, že zadání není jednoznačné – není totiž jasné, který znak uvozovky ukončuje. Uvažujme např.:

```
\uv{\LaTeX{} \verb/tu zpracuje {nespárovanou závorku/}
```

Text je vždy ukončen pravou složenou závorkou, ale nemusí to být ani závorka první, ani závorka párová. Šlo by samozřejmě definovat speciální pravidlo pro makro `\verb`, nicméně problém by se nám opakoval v bledě modrém, pokud uživatel např. definuje:

```
\def\mujverb{\verb}
```

Mimoto by se případná implementace nedokázala vyrovnat s libovolným makrem, které mění kategorii některého znaku. Dokonale by to bylo možné vyřešit jedině tak, že makro `\uv` by provádělo plnou expanzi a postupně provádělo povely hlavního procesoru jeden po druhém (např. zmíněné `\catcode`), přičemž slova by se musela zpracovávat jako jeden celek (kvůli implicitním kernům). Takové makro lze ovšem sotva označit jako rozumně složité.

Nabízí se ještě jedno jednoduché řešení, a to „přemapování“ anglických uvozovek (`‘‘` a `’’`) na české uvozovky. Ani to není ideální. Pro angličtinu totiž probíhá nahrazení těchto dvojic znaků na úrovni fontu (ligatura), zatímco pro češtinu by bylo nutné aktivovat znaky `‘` a `’`. To s sebou v každém případě nese některé nežádoucí důsledky. Pro dvojice znaků je v zásadě možné postupovat dvěma způsoby:

1. definovat první znak jako makro bez parametrů a druhý znak načíst pomocí `\let`, resp. `\futurelet`, nebo
2. definovat první znak jako makro s jedním parametrem (touto cestou se vydala implementace `\declare@shorthand` v novějších verzích Babelu).

V prvním případě se pro načtení druhého znaku provádí příkaz hlavního procesoru, takže nebude fungovat např. následující kód:

`\catcode‘-\active`

Vzhledem k tomu, že právě takovýto kód zapisuje Babel do souboru `.aux` při aktivaci znaku pomocí `\initiate@active@char`, je tento postup nepřijatelný. Se znakem `’` je stejný problém, i když se neprojevuje tak často (uvozuje oktalová čísla v syntaktickém pravidle <*number*>)). Mimoto je zde opět problém s implicitními kerny…

V druhém případě se vše odehrává na úrovni expand procesoru. Tím se odstraní výše uvedené nedostatky, ale problém nastane s mezerami – expand procesor totiž všechny mezery mezi jménem makra a parametrem odstraní, takže například takovýto text: „Ty mrchy furt utíkaj’ do lesa“ se vysází jako: „Ty mrchy furt utíkaj’do lesa.“ Lze to samozřejmě obejít tak, že se mezi znak `’` a následnou mezeru vloží např. sekvence `{}`. Mimoto jsou odsuvníky na konci slov v češtině čím dál vzácnější, takže toto řešení by bylo v zásadě přijatelné. Nakonec jsem se však rozhodl ještě trochu jinak (viz dále).

CSLaTeX umožňuje předefinovat znaky uvozovek prvním způsobem, a to povelem `\csprimeson` (mimochodem, tento název je poněkud zavádějící, protože správná expanze znaků `’` na povely `\prime` v matematickém módu je z hlediska českých uvozovek okrajovou záležitostí). Nový `czech.sty` tento povel nabízí pouze v módu zpětné kompatibility s CSLaTeXem.

Po zvážení všech výše uvedených možností jsem dospěl k názoru, že nejlepší bude vyřešit celou záležitost stejně jako v němčině: pro zápis levé uvozovky používat sekvenci `"‘` a pro zápis pravé uvozovky `"’`. Jednak je znak `"` aktivní i v jiných jazycích (takže se zbytečně neaktivují další znaky), jednak za situace, kdy je beztak potřeba vytvořit novou konvenci, jsem dal přednost řešení, které se používá i jinde. Neaktivní znak `"` je k dispozici v makru `\dq`.

Pro úplnost bych měl dodat, že existuje ještě jedno řešení, které funguje perfektně: upravit tabulku ligatur, aby posloupnost `‘‘` vedla na znak `„` a posloupnost `’’` na znak `“`. Samozřejmě, že takovým fontem pro změnu není snadné vysázet správně anglický text, nehledě na to, že porušuje kompatibilitu s fonty Computer Modern a neumím si představit, že by třeba tyto ligatury Petr Olšák pozměnil v oficiální verzi CS-fontů.

Spojovníky, rozdělovníky a pomlčky

Dalším velkým tématem je zpracování znaku `-`. CSLaTeX definuje dvě makra `\splithyphens` a `\standardhyphens`. Při `\splithyphens` je znak `-` aktivní a snaží se správně rozdělovat slova, která obsahují spojovník. Navíc se snaží neoddělovat příklonku „-li“. Při vší úctě k tvůrcům tohoto makra se sluší poznamenat, že jeho implementace není moc sofistikovaná, protože oddělení příklonky „-li“ předchází tak, že nedovolí zlom v místě spojovníku, pokud za ním následuje písmeno „l“, tj. nezlomí se ani slova jako „česko-latinský“. To je však detail.

Při návrhu nového řešení jsem vycházel z toho, že jediným důvodem, proč by se příklonka „-li“ neměla oddělovat, je příliš malá délka zbývajícího úseku slova na druhém řádku. Obecně jde tedy o vztah mezi délkou slova za spojovníkem a nastavením registru `\righthyphenmin`. Aktivní znak `-` se proto nyní chová tak, že načítá do pomocného makra znaky, dokud jich není alespoň `\righthyphenmin`, nebo dokud nenarazí na konec slova. To znamená, že i slova jako „je-li“ mohou být rozdělena, pokud nastavíte `\righthyphenmin` na hodnotu 2 (či méně). Naopak, i u delších slov může být dělení potlačeno, pokud hodnotu `\righthyphenmin` zvětšíte.

Implementace této funkce není zcela dokonalá, ale měla by postačovat pro většinu situací. Provádí se postupně úplná expanze kódu, přičemž se uvažují znaky, primitiv `\char` a primitivy definované pomocí `\chardef`. Algoritmus se také vyrovná s otevřením a uzavřením skupiny, ale ostatní primitivy považuje za konec slova, takže nezvládne například následující konstrukci:

```
česko-sl\relax ovenský
```

Takto by to samozřejmě nikdo nenapsal, ale umím si představit, že při použití maker mohou podobné situace nastat.

Nové řešení pro Babel také nemění nic na tom, že aktivní znak `-` bude působit problémy jako součást čísel v horizontálním módu a nadále se místo něj musí v těchto případech používat zástupné makro `\minus`.

Na rozdíl od CSLaTeXu je znak `-` aktivní vždy. Bez dalšího nastavení však expanduje na neaktivní znak `-`, takže jeho funkce je naprosto shodná s neaktivní definicí. Pokud chcete aktivovat automatické dělení slov se spojovníkem, musíte buď použít makro `\splithyphens` (stejně jako v CSLaTeXu), nebo uvést jazykový atribut „split“:

```
\languageattribute{czech}{split}
\languageattribute{slovak}{split}
```

Rozdíl je v tom, že pokud je ve stejném dokumentu použita čeština i slovenština, makro `\splithyphens` ovlivňuje oba jazyky, zatímco `\languageattribute` pouze ten z jazyků, pro který je definován. Kombinace obou způsobů v jednom dokumentu může mít nečekané následky.

Na tomto místě je asi vhodné vysvětlit, jak obě makra z CSLaTeXu vlastně fungují. Makro `\splithyphens` je možné použít kdekoli v textu. Pokud je aktuálním jazykem čeština nebo slovenština, znak se rovnou také aktivuje, jinak si CSLaTeX pouze zapamatuje, že aktivace se má provést při nejbližším přepnutí do českého či slovenského jazyka. Pokud je poté jazyk přepnut zpět např. na angličtinu, znak `-` zůstává v CSLaTeXu aktivní. Makro `\standardhyphens` vždy nastavuje kategorii znaku `-` na 12 (ostatní znaky) a vypíná aktivaci znaku `-` při přepnutí na češtinu či slovenštinu.

Nová implementace se v tomto ohledu chová trochu konzistentněji. Makra `\splithyphens` a `\standardhyphens` lze také použít kdekoli a v češtině a slo-

venštině také rovnou mění chování znaku -. V ostatních jazycích si zapamatuje, jaký mód se má použít při dalším přepnutí na češtinu nebo slovenštinu. V každém případě však deaktivuje rozšířenou funkci znaku - pro jiné jazyky. Za zmínku ještě stojí, že nastavení příznaku se neprovádí globálně (stejně jako v CSLaTeXu), takže nastavení `\splithyphens` například uvnitř prostředí `otherlanguage` platí pouze do uzavření této skupiny.

Kódování fontu

Pro začátek si zkuste odpovědět na následující otázku: „Jakým způsobem lze v LaTeXu s NFSS přepínat mezi CM-fonty, EC-fonty a CS-fonty?“ Pokud jste to zatím nevěděli, možná vás trochu překvapí následující kód:

```
\fontencoding{OT1}\selectfont
Použitý font: \fontname\font % cmr10

\fontencoding{T1}\selectfont
Použitý font: \fontname\font % ecrm1000

\fontencoding{IL2}\selectfont
Použitý font: \fontname\font % csr10
```

Problém je v tom, že jádro LaTeX 2_ε počítá pouze s kódováním T1, ale základními fonty pro CSLaTeX jsou CS-fonty. Bylo by sice možné vytvořit variantu CS-fontů s kódováním T1, ale moc bychom si tím nepomohli, protože by bylo nutné změnit také soubor `t1cmr.fd` (standardní součást LaTeX 2_ε) nebo mít dvě verze tohoto souboru, což by zase přinášelo obdobné problémy jako dvě verze `czech.sty` a `slovak.sty`.

Ponechme tedy CS-fonty v kódování IL2 a pro kódování T1 používejme EC--fonty (či dokonce LM-fonty). K tomu je nutné zajistit následující:

1. donutit Babel, aby při inicializaci formátu zavedl vzory dělení pro češtinu a slovenštinu ve dvou kódováních,
2. při přepnutí kódování nastavit správné vzory dělení a upravit `\uccode`, `\lccode` a `\sfcode` a
3. při přepnutí na český, resp. slovenský jazyk nastavit vzory dělení podle právě zvoleného kódování.

Pokud jde o první bod, v Babelu již určitá podpora jiných kódování existovala. Kódování lze zapsat do `language.dat` jako nepovinný parametr oddělený dvojtečkou od jména jazyka. Takto lze v souboru `language.dat` uvést stejný jazyk několikrát pro různá kódování. Aby to fungovalo, je ještě potřeba pro jedno z těchto kódování vytvořit alias bez dvojtečky, tj. výsledek vypadá nějak takto:

```
czech:IL2        czhyph.tex
```

```
czech:T1        czhyph.tex
=czech
slovak:IL2      skhyph.tex
slovak:T1       skhyph.tex
=slovak
```

K tomu je ovšem nutné odstranit ze souborů `czhyph.tex` a `skhyph.tex` explicitní nastavení kódování T1. Kdyby nám nevadilo, že LaTeX bude při startu vypisovat dvakrát „Loading CZ hyphenation patterns…“ a dvakrát „Loading SK hyphenation patterns…“, mohli bychom místo toho použít přímo soubory `czhyphen.tex` a `skhyphen.tex`.

Bohužel to ještě není úplně všechno, protože Babel podporuje pouze kódování fontu, která jsou již zavedena, tj. nenačítá v případě potřeby nové definice ze souboru. Potřebná úprava Babelu není nijak složitá a Johannesi Braamsovi jsem ji poslal již v dubnu, takže doufám, že se brzy objeví i v oficiální verzi. Na okraj poznamenám, že definice kódování vyžadují některá interní LaTeXová makra, pro která Babel nedefinuje žádnou náhradu pro případ, že je použit s formátem plain (např. `\ProvidesFile`). Z toho mimo jiné plyne, že (alespoň prozatím) nebude možné takto Babelem nahradit csplain.

Pokud jde o druhý bod, kódování IL2 je definováno tak, že při přepnutí na IL2 se potřebné kódy změní. Problém však nastává při přepnutí z IL2 do jiného kódování. LaTeX 2_ε v tomto směru moc možností nenabízí, a tak jsem zvolil variantu předefinovat makro `\@@enc@update` tak, aby se v případě, že aktuální kódování je IL2, nejdříve obnovily hodnoty `\uccode`, `\lccode` a `\sfcode` podle T1. Poté se provede změna kódování v LaTeXu pomocí původní definice makra `\@@enc@update` a nakonec se přepnou vzory dělení, pokud jsou pro daný jazyk a kódování zavedeny. Potom už zbývá jen jedna drobná komplikace, a to, že předefinování uc/lc/sf kódů v souboru `il2enc.def` je prováděno nesprávně, ale to lze napravit snadno.

Pokud jde o třetí bod, bylo nutné pozměnit Babel, aby při přepínání vzorů dělení bral ohled i na aktuální kódování. Tato změna se (doufám) také objeví v nejbližší verzi Babelu.

Na závěr ještě dodám, že stejně jako v CSLaTeXu se CS-fonty použijí i v matematickém módu, pokud jsou ovšem zvoleny jako primární fonty dokumentu. Jestli dojde k předefinování matematických fontů, závisí na tom, zda je na začátku dokumentu vybrané kódování IL2, tj. pokud je celý dokument sázen např. s EC-fonty, CS-fonty se do matematického režimu necpou. Při použití pdflatexu to má samozřejmě také vliv na množství fontů přibalených do výsledného PDF.

Co zbývá z CSLaTeXu?

Předcházející text možná vyzněl tak, jako by se teď o vše, co souvisí s českou sazbou, staral jazykový soubor v Babelu a CSLaTeX už nebyl vůbec potřeba. To není úplně přesné. CSLaTeX totiž mimo jiné obsahuje také soubor definice kódování IL2 pro LaTeX (`il2enc.def`) a soubory pro mapování CS-fontů (`il2cm*.fd`). Tyto soubory jsou nezbytně nutné, pokud chcete sázet své dokumenty CS-fonty (to asi chcete), a musí být někde dále spravovány, protože kódování IL2 nebylo do LaTeXu nikdy oficiálně přijato a pokud dokážu posoudit, ani nikdy nebude.

Dalšími soubory, které je třeba spravovat, jsou `czhyph.tex` a `skhyph.tex`, které vytvořil Petr Olšák, aby se i v Babelu mohly používat Ševečkovy vzory dělení (do té doby se tam používaly starší vzory od p. Lhotky). Tyto soubory kupodivu nejsou ani součástí Babelu, ani součástí CSLaTeXu. V distribuci TeXlive se například instalují ze souboru `hyphen-czechslovak.zip`, který (dá-li se věřit hlavičkám) sestavil sám Sebastian Rahtz.

Kompatibilita

Při veškerých úpravách hrála zpětná kompatibilita s CSLaTeXem velkou roli. O něco menší roli pak hrála kompatibilita se stávající podporou češtiny a slovenštiny v Babelu. Konkrétně to znamená, že pokud je definiční soubor jazyka zaveden pomocí `\usepackage{czech}` nebo `\usepackage{slovak}`, použije se mód zpětné kompatibility s CSLaTeXem, který je téměř stoprocentně kompatibilní s CSLaTeXem. Jsou mi známy tyto odlišnosti:

- Aktivní znak `-` vždy respektuje nastavení `\righthyphenmin` a rozděluje i slova, v nichž po spojovníku následuje znak `1`.
- V módu `\splithyphens` nezůstává znak `-` aktivní v jiných jazycích než češtině nebo slovenštině.
- Aktuální datum vkládané pomocí `\today` v CSLaTeXu umožňuje pro češtinu řádkový zlom mezi číslem dne a jménem měsíce, zatímco v nové verzi je zde zlom zakázán. Pro úplnost dodávám, že pro slovenštinu byl na stejném místě zlom řádku zakázán odjakživa.
- Veškerá funkčnost je aktivní až po zavedení `czech.sty`, resp. `slovak.sty`. CSLaTeX používá upravený `hyphen.cfg` a `fonttext.cfg`, takže některé věci jsou zavedené již ve formátu, např. CS-fonty, přepínání vzorů dělení podle zvoleného kódování nebo aktivní znak `-`.
- Nový český a slovenský styl nebude fungovat s LaTeXem 2.09. Tím mám na mysli skutečný LaTeX2.09, nikoli kompatibilní mód LaTeX 2_ε, tj. pokud váš soubor začíná např. nějak takto:

  ```
  \documentstyle[czech]{article}
  ```

je stále možné jej zpracovat s LaTeX 2_ε a novou verzí `czech.sty`. Případné zájemce bych však chtěl upozornit, že tuto variantu jsem testoval jen velmi okrajově.

- Makra `\crq` a `\crqq` ve vertikálním módu nezpůsobují chybu, ale přejdou do horizontálního módu.

Zavedení stylu povelem `\usepackage{czech}` nevypisuje standardní varování Babelu. Je to proto, že v různých (i tištěných) dokumentech se doporučuje zareagovat na toto varování smazáním příslušného souboru `czech.sty`, což by teď problémy spíše přidělalo.

Kvůli kompatibilitě s Babelem jsou zachovány dva povely: `\q` a `\w`. První se používá pro vytvoření kličky (např. u „ď“), druhý pro vytvoření kroužku (např. nad „ů“). V současné době jsou to pouze aliasy pro standardní makra `\v` a `\r`, která jsou patřičně nadefinována pro kódování OT1.

Slovenská podpora v Babelu umožňovala také sázet slovenské znaky pomocí digrafů, např. `"a`↦`ä`, `^o`↦`ô`, apod. Tato možnost je zachována v režimu Babelu, v kompatibilním režimu tyto digrafy nemají žádný speciální význam (tj. sází se jako "a, ˆo, apod.).

Posledním detailem je verze těchto souborů. Poslední babelovský `czech.sty` měl verzi 1.3k, `slovak.sty` verzi 1.3a. Verze souborů z CSLaTeXu měly číslo 2.4. Abych naznačil, že nová verze nahrazuje oba tyto soubory, očísloval jsem ji jako 3.0. Aktuální verze (s jazykovými atributy a opravami několika chyb) má pak číslo 3.1.

Přehled

Následující tabulka přehledně shrnuje funkce nových českých a slovenských stylů. Vlevo je kód pro kompatibilní mód (CSLaTeX), vpravo pro přirozený mód (Babel).

• inicializace s CS-fonty:

`\usepackage{czech}`	`\usepackage[czech]{babel}` `\usepackage[IL2]{fontenc}`

• inicializace s EC fonty (výhledově LM fonty):

`\usepackage[T1]{czech}`	`\usepackage[czech]{babel}` `\usepackage[T1]{fontenc}`

• uvozovky bez `\verb`:

`\uv{text v uvozovkách}`	`\uv{text v uvozovkách}`

• uvozovky kolem `\verb`:

`\clqq \verb|verbatim text|\crqq`	`"‘\verb|verbatim text|"’`

• přemapování anglických uvozovek na české:

`\csprimeson`	`(není implementováno)`

• francouzské uvozovky:

```
\flqq text\frqq
```

```
"<text">
```

• použití staré definice makra `\uv`:

```
\usepackage[olduv]{czech}
```

```
(není implementováno)
```

• aktivní znak `-`:

```
\usepackage[split]{czech}
```

```
\usepackage[czech]{babel}
\languageattribute{czech}{split}
```

• (dočasná) aktivace znaku `-` kdekoli v textu:

```
\splithyphens
```

```
\splithyphens
```

• (dočasná) deaktivace znaku `-` kdekoli v textu:

```
\standardhyphens
```

```
\standardhyphens
```

• správně rozdělený spojovník bez aktivního znaku `-`:

```
(není implementováno)
```

```
česko"=slovenský
```

• záporné číslo v horizontálním módu při `\splithyphens`:

```
\kern\minus1pt
```

```
\kern\minus1pt
```

• české dělení bez přeložených textových řetězců:

```
\usepackage[nocaptions]{czech}
\begin{document}
český text, anglické datum: \today
\end{document}
```

```
\usepackage[czech,english]{babel}
\begin{document}
\begin{otherlanguage*}{czech}
český text, anglické datum: \today
\end{otherlanguage*}
\end{document}
```

• slovenské znaky pomocí digrafů:

```
(není implementováno)
```

```
m"ak^cene zo znakov ASCII
```

• slovenské digrafy s čárkou:

```
(není implementováno)
```

```
\usepackage
  [activeacute,slovak]{babel}
\begin{document}
teraz aj d'l^zne zo znakov ASCII
```

Babylón mluví hindsky

Zdeněk Wagner

Babel poskytuje jednotné rozhrani pro tvorbu vícejazyčných dokumentů. V současné době není bohužel podporován žádný z indických jazyků. Sazba v indických jazycích je založena na specializovaných balíčcích. Nejpokročilejším z nich je Velthuis Devanāgarī for TEX, protože již obsahuje hindské nadpisy i makro pro tisk data v evropském stylu. Definiční soubor pro začlenění hindštiny do babelu byl tedy v sooučasné době vyvinut.

Druhá část článku vysvětluje rozdíl mezi UNICODE a Velthuisovou transliterací. To je důležité pro pochopení nástroje sloužícího ke konverzi hindských a sanskrtských dokumentů z MS Wordu a OpenOffice.org do TEXu pomocí procesoru XSLT 2.0 a perlovského skriptu, jakož i metody, jak lze vytvořit prohledávatelný PDF soubor.

Závěrem článek diskutuje možnosti dalšího vývoje, zejména možnosti, jež nabízí XETEX a integrace jazyka LUA do pdfTEXu.

Práce byla prezentována na konferenci TUG 2006 v Marrákéši a zveřejněna v angličtině v TUGboatu [13]. Český překlad je publikován se souhlasem organizátorů konference i redakce TUGboatu.

Úvod

Balíčky pro sazbu v různých indických jazycích jak pro plain TEX, tak pro LATEX, jsou dostupné na CTAN již dlouhou dobu. Autoři těchto balíčků věnovali pozornost podpoře písma. To obsahuje potíže, jež nelze řešit TEXem samotným. Vytváření spřežek a vkládání nesamostatných samohlásek (máter) do subskriptů a superskriptů lze řešit pomocí ligatur v souborech TFM. Forma spřežky však závisí též na jazyce. Zatímco v sanskrtu a tradiční hindštině se užívají spřežky क्त (kta) a न्न (nna), v moderní hindštině jsou obvykle nahrazeny polovičními formami क्त a न्न. To lze řešit pouze preprocesorem svázaným s balíčkem (La)TEXových maker.

Balíčky používají indologové z celého světa i lidé v Indii. Je tedy nešťastné, že balíčky podporují pouze písmo a chybí jim podpora jazyka. Výjimkou je Velthuis devanāgarī for TEX [1]. Počínaje verzí 2.13 obsahuje definice nadpisů i evropského stylu data a nabízí makra umožňující přepínání mezi nimi a anglickou verzí. Nabízí tedy mini-babel, jaký obsahuje CSLATEX. Dalším přirozeným krokem je tedy vytvoření definičního souboru jazyka, jímž bude hindština integrována do Babylónu.

Stvoření definičního souboru jazyka

Cílem práce bylo umožnit transparentní použití hindštiny ve vícejazyčných dokumentech pomocí `\usepackage[hindi]{babel}`. Příprava definičního souboru jazyka pro babel není obtížná. Je nutné definovat makra s názvy nadpisů jako `\chaptername` a makro `\today` pro tisk data. Tyto definice již byly v balíčku přítomny a bylo též vyřešeno přepínání písmových variant Bombay, Calcutta a Nepali. Jejich pouhé vložení do definičního souboru jazyka však není postačující. Zobrazení textů v dévanágarském písmu vyžaduje řadu speciálních maker. Jak již bylo zmíněno dříve, hindský text nemůže být předložen TEXu přímo, ale je nutné předchozí zpracování preprocesorem. Bylo by neúčelné, kdyby makra ze souboru `devanagari.sty` byla překopírována do definičního souboru jazyka a pak musela být udržována na dvou místech. Definiční soubor jazyka tedy načítá `devanagari.sty`, jehož parametry jsou definovány jako jazykové atributy. Ve skutečnosti tento definiční soubor bez fontů a preprocesoru stejně nebude fungovat. Požadavek, že musí být instalován kompletní balíček Velthuis devanāgarī for TEX, tak nepředstavuje žádné omezení.

Balíček `devanagari.sty` obsahuje makra pro nadpisy a tisk data v angličtině. Tyto definice nelze odstranit, protože jsou dokumentovány a jejich odstranění může poškodit existující dokumenty. Na druhou stranu však tyto definice budou kolidovat s jádrem babelu. Balíček byl tedy upraven tak, aby tato makra byla definována pomocí `\providecommand` a definice je odložena použitím `\AtBeginDocument`. Tím je zajištěno, že zmíněná makra existují, ale definice z babelu mají přednost nezávisle na pořadí, v němž se balíčky načítají. Definiční soubor jazyka kontroluje verzi balíčku a stěžuje si, je-li instalována stará verze. Jak již bylo zmíněno, nestačí aktivovat hindštinu jako hlavní jazyk nebo použitím prostředí pro změnu jazyka. Text musí být uzavřen ve skupině {`\dn` ... }, jinak jej preprocesor nenajde.

UNICODE vs. Velthuisova transliterace

Devanāgarī má původ ve starém písmu Brāhmī. Patří mezi abugidy. Každá souhláska (vyanjana) obsahuje inherentní samohlásku (v písmu devanāgarī *a*) a ostatní samohlásky jsou přidávány jako diakritická znaménka v blízkosti souhlásky. Skupiny souhlásek často tvoří spřežky, jen menšina z nich se zapisuje pomocí znaku viráma. Počáteční forma samohlásek má jiný tvar než samohlásková diakritická znaménka (nesamostatné samohlásky, mātrā).

UNICODE je založen na znacích. Inherentní samohláska *a* se nezapisuje. Znak U+0915 označuje tedy slabiku क (ka). Slabika कि (ki) je reprezentována dvěma znaky UNICODE U+915 U+93F. Záměna pořadí znaků v tištěném výstupu je přenechána jako úloha zobrazovače [2]. Samostatné samohlásky

(počáteční formy) mají odlišné kódy, tj. kód इ (i) je U+0907. Trojice znaků U+0915 U+094D U+0924 označuje sanskrtskou spřežku क्त (kta). Tento znak však nemusí být přítomen v moderních fontech pro hindštinu. V takovém případě bude zobrazeno क्त. Chceme-li zobrazovač přinutit k vytvoření této formy i v případě, kdy font sanskrtskou spřežku obsahuje, musíme vložit znak zero-width-joiner. Kód v UNICODE pak bude U+0915 U+904D U+200D U+0924.

Transliterační schéma, které vyvinul Frans Velthuis, se snaží co nejvíce přiblížit školním praktikám. Devanāgarī je tradičně přepisováno latinkou, přičemž dlouhé samohlásky, cerebrální souhlásky a nosovky jsou označovány diakritickými znaménky [10]. Forma se mírně liší v různých učebnicích a slovnících. Velthuisova transliterace je sedmibitovým kódováním, takže lze znaky psát na běžné americké klávesnici. Je založena na výslovnosti, přestože v hindštině se uprostřed mnoha slov a na konci slov inherentní *a* nevyslovuje. Inherentí *a* uprostřed slov se vždy musí zapsat, ale koncové *a* obvykle vynecháváme. Slovo करना tedy musíme psát jako *karanaa*, zatímco slovo घर píšeme jen *ghar*.

Důležitost rozdílu mezi těmito přístupy bude ukázána v následujícím textu.

Konverze do Velthuisovy transliterace

Příprava knih je často společnou prací autorů, editorů a sazeče. Autoři zřídka dodají text v TEXu, často používají jiné textové editory, většinou MS Word. První úlohou je tedy konverze dodaného rukopisu do TEXu. Téměř veškeré značkování musí být odstraněno a nahrazeno jiným, které odpovídá grafickému návrhu. S výhodou lze otevřít soubor v OpenOffice.org a uložit jej v nativním formátu, jímž je XML. Ačkoli jsou dostupné různé konverzní nástroje, jejich výstup stále zachovává příliš mnoho formátování. Využití TEXML by si též vyžádalo příliš mnoho práce, kterou by stěží bylo možno použít v dalších knihách. Protože dokument v OpenOffice.org je uložen v XML, lze konverzi provést pomocí XSLT. Jednoduchý styl může odstranit veškeré formátování a zachovat tučné písmo, kurzívu a poznámky pod čarou. Některé znaky UNICODE však nejsou přímo dostupné. Naštěstí Saxon v. 8.x [5], jenž implementuje XSLT 2.0, nabízí znakové mapy, jejichž pomocí lze znaky konvertovat na TEXové sekvence. To však není pro texty v devanāgarī postačující. Pokud bychom pouze převedli znaky z UNICODE na odpovídající latinková písmena, všechna inherentní *a* by byla ztracena.

Výstup transformace XSLT lze předložit TEXu. Přesto je doporučeno jisté následné zpracování. Někdy je tučný text vložen tak, že je každé písmeno ztučněno samostatně. Spojení do jediného makra `\textbf` dosáhneme snáze v perlu než v transformačním stylu. Navíc je celý dokument v OpenOffice.org uložen na jednom řádku. Protože potřebujeme výsledný dokument editovat ručně, je vhodné je nalámat na řádky vhodné délky. Bez jakéhokoliv programování toho docílíme modulem `Text::Wrap`. Konverzi lze tedy docela snadno rozdělit mezi XSLT a perl.

Velthuisova transliterace kóduje dlouhé samohlásky buď zdvojením, nebo pomocí verzálek. Transformační styl používá vždy verzálky, aby se předešlo nejednoznačnostem. Například कई bude převedeno na *kaI*, protože *kaii* by se zobrazilo jako कैइ, což je špatně. Problém by se dal též řešit vložením prázdných složených závorek, ale použití verzálek je jednodušší. Samostatné samohlásky jsou transformovány přímo na odpovídající písmena. Před nesamostatné samohlásky je vloženo rovnítko. Souhlásky jsou následovány rovnítkem a znak viráma je transformován na podtržítko. Pak přichází na řadu perlovský skript. Zpočátku je každý odstavec samostatným řádkem, jenž musí být zpracován. První úlohou je vytvoření spřežek. To bude správně fungovat jak v sanskrtských slovech, tak v moderní hindštině, kde se některé spřežky neužívají, např. ve slově अड्डा. Znak viráma pak vloží preprocesor. Spřežky budou vytvořeny příkazem

```
while (s/(\{\\dn [^}]*)=_/$1/) {}
```

V následujícím kroku jsou doplněny nesamostatné samohlásky. Dvojice rovnítek jsou odstraněny a osamocená rovnítka označují inherentní *a*, jež musí být vložena. Je toho docíleno těmito řádky

```
while (s/(\{\\dn [^}]*)==([aAiIuU.eo])/$1$2/){}
while (s/(\{\\dn [^}]*)=/$1a/) {}
```

Pokud nekonvertujeme sanskrtský text, odstraníme ještě koncové inherentní *a*.

```
while (!$opt_sanskrit && s/(\{\\dn [^}]*)a([^.a-zA-Z])/$1$2/) {}
```

Nakonec přelámeme řádek.

```
eval {
  print wrap(’’, ’’, $_);  1;
} or do {
  warn "Warning: $@";  print;
};
```

Prohledávatelné PDF soubory

PDF soubory mají důležitou roli jako elektronické dokumenty. Názvy kapitol mohou být vloženy do záložek a příbuzné části lze provázat hypertextovými odkazy. Přesto je žádoucí, abychom byli schopni vyhledávat slova a věty. Indická písma zde přinášejí velký problém. PDF, podobně jako PostScript, zobrazuje glyfy, ale vstupní pole ve vyhledávacím dialogu očekává znaky v UNICODE. Soubory vytvořené balíčkem `devnag` jsou tedy neprohledávatelné. XƎTEX [14] může používat fonty OpenType, avšak ukazuje se, že ani takto vytvořené soubory prohledávatelné nejsou. Situace není jednoduchá. XƎTEX může používat různé stroje pro vytvoření PDF a výsledek je podle toho různý. OpenOffice.org je nepatrně úspěšnější, protože všechna jednoduchá slova, která neobsahují spřežky, jsou vyhledatelná.

Klíčový problém plyne z rozdílu mezi glyfy a znaky. Mapování mezi glyfy a znaky lze vložit do mapy ToUnicode. Tato vlastnost je již implementována

v balíčku `cmap.sty`. Experimentální mapa ToUnicode byla proto vytvořena pro balíček Velthuis Devanāgarī. Protože pdfTEX vkládá tyto mapy podle kódování fontu, bude každý indický fomt vyžadovat vlastní jméno kódu pro LATEX. V současnosti Devanāgarī, Bengali a Gurmukhi používají kódování U a spoléhají na preprocesor s obdobnou funkcionalitou. V tomto experimentálním projektu je definováno kódování X0900, což se vztahuje k příslušnému bloku v UNICODE.

Fonty `dvng` obsahují jednoduché znaky, spřežky a části, které se slepují dohromady TEXovými makry. Jednoduché znaky a spřežky jsou mapovány přímo na znaky UNICODE. Poloviční formy souhlásek jsou mapovány na příslušné znaky následované virámem. Vattu je přidáváno k úplné souhlásce. Je proto mapováno na र (ra), před nímž je přidán znak viráma. Tím jsou znaky se spřežkami vyhledatelné.

Všechny problémy bohužel vyřešit nelze. Protože PDF pracuje s glyfy, jsou nesamostatné samohlásky *i* vloženy před souhlásky. Vyhledáváme-li taková slova, je nutno zapsat je do vstupního pole ve vyhledávacím dialogu tak, jak vypadají graficky, tj. musíme psát nesamostatné *i* před souhláskou. Acrobat Reader je často zmaten samohláskou umístěnou nad nebo pod souhláskou i znakem *vattu*. Vytvoří se tím nadbytečná mezislovní mezera. Hledáme-li tedy कुल्लू, musíme do vstupního pole zapsat dvě slova कु ल्लू. Slovo ड्राइवर musíme zapsat způsobem, jenž je ve Velthuisově systému nemožný, a to jako dvě slova ड् राइवर. UNICODE povoluje zapsat na začátek slova nesamostatnou samohlásku, což je, samozřejmě, nesprávné. Díky této chybné vlastnosti jsou uvedená slova vyhledávatelná. Nadbytečná mezera se nevytvoří, pokud je slabika kreslena jako jeden glyf ve fontu. Slova रुकना a मात्रा lze najít v podobě, jak jsou zapsána. Podrobnější informace jsou uvedeny v dokumentaci experimentální mapy ToUnicode [12].

Mapa ToUnicode se používá též při kopírování textu ze souboru PDF do jiné aplikace. Je tedy pochopitelné, že nastávají stejné komplikace. Vložený text má nadbytečné mezery za znakem vattu a za krátkou i dlouhou nesamostatnou samohláskou *u*. Slova s krátkým nesamostatným *i* budou vizuálně vypadat dobře, ale jejich reprezentace v UNICODE bude špatná. Pokud slovo दिल्ली zkopírované z PDF bude posláno do řadicího algoritmu, objeví se na nesmyslném místě, protože program najde slovo začínající nesamostatnou samohláskou *i*, což hindský pravopis nepřipouští.

Budoucí vývoj

Největší nevýhodou současného balíčku Velthuis devanāgarī for TEX je nutnost použití preprocesoru. Pokud připravujete dokument v jediném jazyce, nemusí to připadat tak nepohodlné. Sázíte-li však trojjazyčný dokument v hindštině, bengálštině a pandžábštině, musíte zpracovat zdrojový soubor postupně třemi preprocesory. Preprocesor DEVNAG umí zpracovat několik pandžábských a bengál-

ských slov v hindském odstavci použitím úhlových závorek, ale všechny preprocesory nemají tak pokročilé schopnosti. Z nutnosti použití preprocesoru vyplývají těžkosti při tvorbě rejstříku, jež nebyly dosud vyřešeny. Náhrada preprocesoru jiným mechanismem je tedy důležitým krokem.

První myšlenkou byla reimplementace preprocesoru v encTEXu [6]. Ten umí konvertovat vstupní znaky na libovolné tokeny podle konverzní tabulky. Konverzi lze zapínat a vypínat změnou hodnoty `\mubytein`. Konverzní tabulka může též ovlivnit zápis do souboru primitivem `\write` podle hodnoty v registru `\mubyteout`. Protože konverze probíhá na znakové úrovni ve vstupním procesoru, nelze odlišit znaky ve slovech od znaků v řídicích sekvencích. Reimplementace preprocesoru v encTEXu by tedy nebyla snadná a výsledný kód by nebyl efektivní.

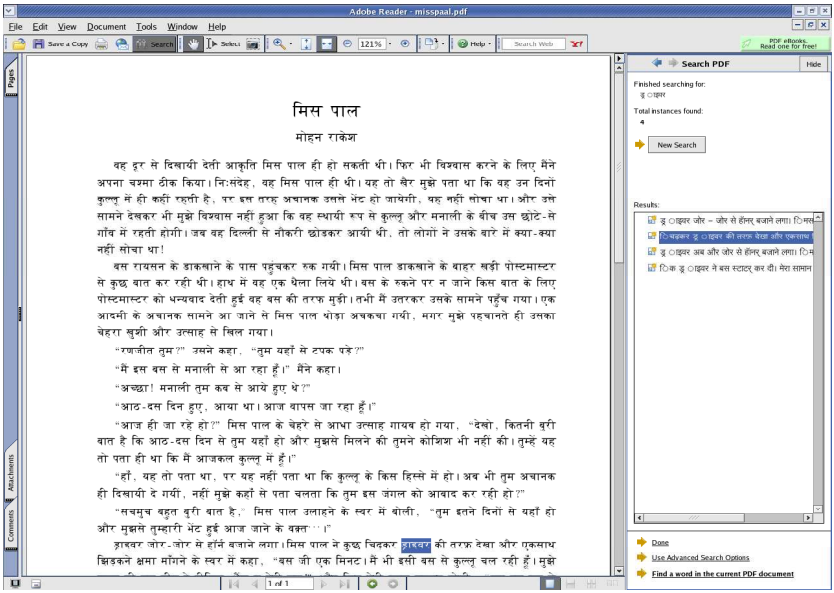
Slibnější možnost nabízí integrace skriptovacího jazyka LUA [3] do pdfTEXu. Preprocesor lze reimplementovat v jazyku LUA, a navíc lze přidat další vylepšení. Bude možno číst texty jak ve Velthuisově transliteraci, tak v UNICODE. Bude též možno vysázet dokumenty, které již byly zpracovány preprocesorem, takže kompatibilita nebude ztracena. Jakmile bude LUA integrován do XƎTEXu, bude možno volit mezi fonty `dvng` a OpenType. Pak bude též snazší implementovat software pro tvorbu rejstříků v indických jazycích.

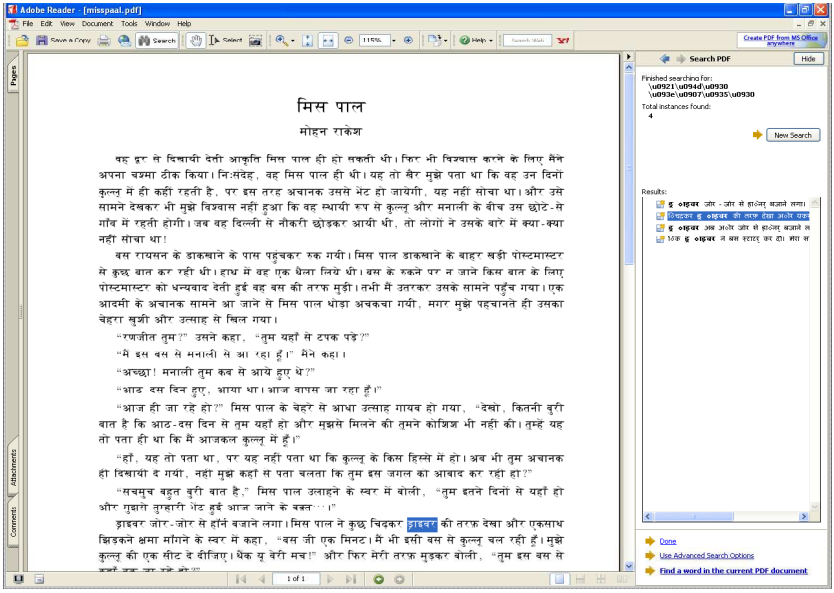
Požadavky na vícejazyčné prostředí

Vícejazyčné prostředí není vyžadováno jen v TEXu, ale v celém operačním systému. Především je nutné zobrazit správně všechny znaky UNICODE. Skupiny souhlásek s virámy je nutno správně spojit do spřežek a nesamostatné samohlásky přesunout na správná místa. V Linuxu k tomu slouží například knihovny ICU [4] a Pango [8]. Tyto knihovny však nejsou dosud používány všemi programy. Ani Firefox nepoužívá Pango ve standardní konfiguraci, ale musí být použití aktivováno nastavením `MOZ_PANGO_ENABLE=1`. MS Internet Explorer zobrazuje hindské texty korektně.

Zdrojový text musí též být připraven v UNICODE. Zde je stále problém s editory v Linuxu. OpenOffice.org, gEdit a <oXygen/> (XML editor) [7] pracují správně. Nezkoušel jsem yudit, ale podle referencí jiných uživatelů je prý též s indickými jazyky funkční. Podpora indických písem v jiných editorech stále chybí.

Vstupní text v indických písmech musí být přijímán a správně zobrazován ve všech aplikacích. Zde má problém Adobe Acrobat Reader ve Windows. Pomocí hindské klávesnice se mi nepodařilo napsat nic do vstupního pole vyhledávacího dialogu. Sice se cosi podivného zobrazilo, ale hledání vůbec nefungovalo. Text může být přenesen z jiné aplikace pomocí copy&paste, ale zobrazí se v entitách UNICODE. Srovnání verzí pro Linux a Windows je uvedeno na obrázku 1. Verze pro Linux je uživatelsky pohodlnější, ale obě vyhledávají stejně dobře.





Obrázek 1: Vyhledávání slova ड्राइवर (vloženého jako ड् ाइवर) v Adobe Acrobat Readeru 7 v Linuxu (nahoře) a ve Windows XP (dole).

K plné funkčnosti vyhledávání a kopírováni textů v PDF je nutno upravit specifikaci CMap. V současnosti jsou možná mapování 1:1 a 1:mnoho. Abychom mohli měnit pořadí glyfů a správně seřadit vícedílné samohlásky v drávidských jazycích, potřebujeme mapování mnoho:mnoho. Pro lepší pochopení je v tabulce 1 ukázáno několik dévanágarských a malajálamských slabik.

Tabulka 1: Vybrané dévanágarské a malajálamské slabiky

Význam	**Devanāgarī** देवनागरी	**Malayālam** മലയാളം
ma	म	മ
maa	मा	മാ
mi	मि	മി
mii	मी	മീ
me	मे	മെ
mo	मो	മോ

Závěr

V článku je ukázáno, jak byl vytvořen babelovský modul pro hindštinu. Dále se autor zamýšlí nad možnostmi použití indických textů v elektronických dokumentech. Jsou též popsány nástroje pro konverzi souborů z MS Wordu a OpenOffice.org do TEXu a pro tvorbu prohledávatelných souborů PDF. Nástroje jsou dostupné z autorovy webové stránky [11].

Poděkování

Autor děkuje ostatním vývojářům Velthuis Devanāgarī for TEX, protože zmíněný balíček je základem tohoto projektu, jakož i Karlu Píškovi za konverzi indických fontů do formátu Type 1 [9]. Zvláštní poděkování zaslouží Anshuman Pandey za překlad nadpisů do hindštiny a John Smith a Arnošt Štědrý za poskytnutí testovacích souborů vytvořených XƎTEXem. Autor dále děkuje Alexandru Babičovi za provedení testů v Ubuntu a Petru Tomáškovi za vysvětlení způsobu zobrazení písem v Xorg. Za finanční podporu účasti na konferenci TUG 2006 autor děkuje CSTUGu a TUGu.

Reference

[1] Devanāgarī for TEX. `http://devnag.sarovar.org/`.

[2] Joan Aliprand et al. *The Unicode Standard*, chapter South Asian Scripts. The Unicode Consortium, 2003. http://www.unicode.org/faq/indic.html#5.
[3] Hans Hagen. LuaTEX: Howling to the moon. *TUGboat*, 26(2):152–157, 2005. http://www.tug.org/TUGboat/Contents/contents26-2.html.
[4] International components for Unicode. http://icu.sourceforge.net/.
[5] Michael Kay. Saxon, the XSLT and XQuery processor. http://saxon.sourceforge.net/.
[6] Petr Olšák. encTEX. http://www.olsak.net/enctex.html.
[7] <oXygen/> xml editor. http://www.oxygenxml.com/.
[8] Pango. http://www.pango.org/.
[9] Karel Píška. Indic Type 1 fonts for TEX. CTAN:fonts/ps-type1/indic.
[10] Transliteration pages. http://homepage.ntlworld.com/stone-catend/translit.htm.
[11] Zdeněk Wagner. Můj volně šiřitelný software. http://icebearsoft.euweb.cz/sw.php.
[12] Zdeněk Wagner. Prohledávatelné pdf s dévanágarskými texty. http://icebearsoft.euweb.cz/dvngpdf/.
[13] Zdeněk Wagner. Babel speaks Hindi. *TUGboat*, 27(2):176–180, 2006.
[14] The XƎTEX typesetting system. http://scripts.sil.org/xetex.

Summary: Babel Speaks Hindi

Babel provides unified interface for creation of multilingual documents. Unfortunately none of Indic languages is currently supported. Typesetting in Indic languages is based on specialised packages. The most advanced from them is Velthuis Devanāgarī for TEX because it already provides Hindi captions as well as a macro for a Europian style date. A language definition file for plugging Hindi into babel has therefore been recently developed.

The second part of the paper explains differences between UNICODE and Velthuis transliteration. This is important for understanding the tool that can convert Hindi and Sanskrit documents from MS Word and OpenOffice.org into TEX via an XSLT 2.0 processor and a Perl script as well as a method of making the PDF files searchable.

Finally the paper discusses some possibilities of further development, mainly the advantages offered by XƎTEX and by forthcoming integration of LUA into pdfTEX.

Babylón v TEX Live 2007

Zdeněk Wagner

V příloze tohoto čísla Zpravodaje najdete DVD TEX Live 2007. V této verzi ještě není obsažen nový modul pro češtinu a slovenštinu, o němž jste si mohli přečíst v článku Petra Tesaříka. Musíte si jej nainstalovat sami, a přitom zjistíte, že to podle návodu jaksi nejde. TEX Live 2007 totiž má jednu novinku, jíž je XETEX. A kvůli němu došlo k drobné změně v `language.dat`. Instalaci nové češtiny a slovenštiny je tedy nutno provést nepatrně jinak.

Návod Petra Tesaříka může na méně zkušené uživatele působit poněkud komplikovaně. Návod je totiž psán obecně pro jakoukoliv distribuci TEXu, takže začátečník si s tím nemusí poradit. A pokud chcete češtinu a slovenštinu instalovat do více počítačů, například do stolního počítače doma, do počítače v zaměstnání a do notebooku, nějaký automatický skript by se hodil. Takový skript byl vytvořen a funguje nejen v TEX Live, ale i v teTEXu (včetně verze pro OS/2 resp. eComStation) a v MiKTEXu. Instalační skript zajistí též funkčnost češtiny a slovenštiny v XELATEXu. Odkaz na instalační balíček najdete na `www.cstug.cz` v rubrice software.

Po instalaci češtiny a slovenštiny do Babylónu je nutno znovu vygenerovat formáty. Pokud čtete tento článek ještě před instalací TEX Live, máte jistou výhodu. TEX Live totiž při instalaci formáty nevytváří, ale pouze nakopíruje datové soubory a připraví konfiguraci. Formát se vygeneruje automaticky při jeho prvním použití. Pokud nainstalujete češtinu a slovenštinu (a případně hindštinu) ihned po instalaci TEX Live, nemusíte se o generování formátů starat.

Po instalaci TEX Live si nastavíme `PATH` tak, jak je to instalačním skriptem navrženo. Pak rozbalíme instalační balíček češtiny, slovenštiny a hindštiny. Instalační skript není kumulativní, je nutno všechny jazyky nainstalovat najednou. Pokud nainstalujete jen češtinu a slovenštinu budete chtít doinstalovat později, budete muset nejprve celý balíček odinstalovat a potom oba jazyky nainstalovat znovu, což může vést za jistých okolností k problémům. Podrobnější popis je v manuálu dodávaném s balíčkem. Pokud instalujeme do čerstvě nainstalované distribuce TEX Live, nebudeme generovat formáty.

Poté se podíváme do konfigurace formátů. Uživatelé nových linuxových distribucí, kteří mají v *locales* nastaveno kódování UTF-8, pravděpodobně budou chtít aktivovat encTEX [1], aby i české písmeno bylo pro TEX obyčejným znakem kategorie 11 a nemuselo se číst s využitím aktivních znaků po částech. Při použití encTEXu totiž i česká a slovenská písmena s diakritikou fungují správně v konstrukcích s `\futurelet` a v podmínkách `\if`. EncTEX zapneme v (pdf)LATEXu parametrem `-enc`, ale nesmíme změnit defaultní kódování, to musí zůstat cp227.

V opačném případě se nenačtou bezchybně vzory dělení pro další jazyky. Po spuštění programu TEXCONFIG a volbě položky „Formats“ se otevře v editoru soubor `fmtutil.cnf`. Konfigurace (pdf)LaTeXu se zapnutým encTeXem vypadá např. takto:

```
latex pdfetex language.dat -enc -default-translate-file=cp227.tcx *latex.ini
pdflatex pdfetex language.dat -enc -default-translate-figure=cp227.tcx *pdflatex.ini
```

Podobně můžete aktivovat encTeX pro CSplain. Postupujte podle komentářů v souboru `fmtutil.cnf`. Tím je instalace hotova a můžete začít psát dokumenty.

Se zapnutým encTeXem lze volit vstupní kódování jak pomocí TCX tabulek, tak novými primitivy encTeXu. Ty využijeme zejména v případě, kdy je vstupní soubor zapsán v kódováni UTF-8. Použijete-li fonty v kódováni IL2 (CS-fonty), pak český dokument bude začínat řádky:

```
\input csenc-u
\documentclass{article}
\usepackage[IL2]{fontenc}
\usepackage[czech]{babel}
```

Soubor `csenc-u.tex` je součástí encTeXu a najdete jej v TeX Live. Chcete-li použít fonty v kódování T1, změní se jen dva řádky:

```
\input csenc-u-t1
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[czech]{babel}
```

Soubor `csenc-u-t1.tex` včetně instalačního návodu najdete na WWW serveru autora.

Poznámka: pro funkčnost hindštiny v Babylónu potřebujete nejnovější verzi balíčku DEVNAG z CVS [2].

Literatura

1. Petr Olšák: Nový encTeX – kódování UTF-8 v TeXu. Zpravodaj Československého sdružení uživatelů TeXu, **13** (2), 98–106 (2003).
2. Devanāgarī for TeX. `http://devnag.sarovar.org/`

Summary: Babel in TeX Live 2007

With inclusion of XeTeX into TeX Live the structure of `language.dat` has been changed slightly. Due to this fact the new Czech and Slovak module, which is

not yet distributed with official Babel, cannot be installed smoothly. The article introduces an installation package of the new module not only for TEX Live but also for other famous TEX distributions. Functionality of XƎLATEX is also preserved.

Abecední řazení a sestavování rejstříků výhradně pomocí makrojazyka TEXu

Petr Březina

Ve světě TEXu se pro abecední řazení rejstříků obvykle užívá externích programů, jako je např. makeindex. TEX ovšem poskytuje silný makrojazyk, který umožňuje vytvořit mechanismus schopný kvalitně provádět řazení podle abecedy. Hlavní výhoda využití takového mechanismu pro tvorbu rejstříků by spočívala v tom, že dokument by bylo možno zpracovat včetně rejstříku na každém počítači, kde je nainstalován TEX, nezávisle na dalším programovém vybavení (externí programy nemusí být všude k dispozici, mohou být závislé na použitém operačním systému atd.). Přesto se v praxi této možnosti většinou nevyužívá. V jedenácté kapitole knihy *Typografický systém TEX* [3] autor vyjmenovává několik důvodů: „Algoritmus by byl velmi těžkopádný, zatěžoval by paměť TEXu, zdržoval by formátování a v neposlední řadě úpravy algoritmu pro třídění podle národních abeced by byly velmi obtížné.“ Tato nedůvěra ve schopnosti TEXu se pro mě stala výzvou, abych se pokusil v makrojazyku TEXu napsat prakticky využitelný program na abecední řazení.

Při tvorbě makra sort, jak jsem nakonec nazval výsledek svého úsilí, jsem se inspiroval programem csr Petra Olšáka [cf. 2], odkud jsem převzal myšlenku čtyřprůchodového abecedního řazení a myšlenku třídicí tabulky. Posléze jsem k tomuto makru udělal soubor obslužných maker na tvorbu rejstříku; při návrhu vzhledu rejstříku jsem nalezl inspiraci ve článku *An indexing facility for TEX* [4]. Celý makrobalíček, který dostal název index, si čtenář může stáhnout z mých internetových stránek na adrese http://www.volny.cz/petr-brezina/.

Poté co jsem dokončil makrobalíček index, jeví se mi výše citované argumenty jako nepřesvědčivé. Pokusím se proto v krátkosti zhodnotit výhody a nevýhody svého řešení v konkurenci s externími programy. Jedna významná přednost makrobalíčku index zazněla již v prvním odstavci tohoto článku; na tomto místě ji už opakovat nebudu a přejdu k dalším vlastnostem. Řadicí algoritmus lze pomocí třídicí tabulky snadno přizpůsobit pro různé jazyky; navíc je celé makro sort pouze TEXovský kód, takže si ho šikovný TEXista může vždycky přizpůso-

bit konkrétním potřebám (např. přidat pátý průchod, bude-li to nějaký jazyk vyžadovat). Takovou flexibilitou se standardně užívaný program pro rejstříky makeindex pochlubit nemůže; doslechl jsem se však o programu xindy, který prý umožňuje definovat pravidla abecedního řazení pro různé jazyky. Paměťové nároky makra sort nejsou nijak zvlášť velké a jsou v podstatě nezávislé na délce seznamu hesel, která mají být seřazena, takže lze pomocí tohoto makra třídit mnohatisícové rejstříky i ve starém dobrém emTEXu. Největší slabinou makra sort je jeho rychlost, která je mnohonásobně menší než rychlost programu makeindex; na druhé straně rychlost makra sort je srovnatelná s rychlostí programu csr. I když činnost makra je poměrně pomalá, nebylo by správné příliš zdůrazňovat, že zdržuje formátování, neboť rejstřík se obyčejně negeneruje při každém formátování v průběhu vzniku dokumentu, nýbrž až nakonec, když už je celý dokument napsán. Je také zapotřebí uvědomit si, že při použití makrobalíčku index lze vytvořit rejstřík během jediného zpracování dokumentu TEXem, zatímco při použití externího programu je pro vytvoření rejstříku nutno spustit TEX dvakrát a jedenkrát spustit onen externí program. Zdá se mi, že makrobalíček index nabízí propracovanější způsob zpracování čísel stran v rejstříkových heslech než program makeindex. Na tomto místě se sluší upozornit na jedno nebezpečí, které číhá na uživatele československé mutace programu makeindex, známé pod jménem csindex: všiml jsem si, že linuxová verze programu csindex řadí chybně plankton < plaňka, zatímco v DOSu jsou obě tato slova řazena správně; chybu jsem nahlásil autorům linuxové verze programu csindex.

Technické souvislosti aneb Trocha historie

Myslím si, že nebude nezajímavé nahlédnout do „strojovny“ makra sort a povědět si, jakým způsobem pracuje. Od prvního návrhu po konečný výsledek prodělalo toto makro zásadní proměnu. Při tvorbě makra jsem věnoval obzvláštní péči jeho rychlosti; nejsem však matematik ani informatik a nevím, zda se mi podařilo nalézt optimální řešení.

Vstupní soubor je nejprve podle primární řadicí platnosti rozdělen na šestnáct pomocných souborů. Jejich velikost je různá v závislosti na tom, kolik položek začíná jakým písmenem; kdyby např. všechny položky začínaly písmenem a, jeden pomocný soubor by obsahoval všechny položky, zatímco zbylých patnáct souborů by bylo prázdných, takže bychom si tím dělením vůbec nepomohli. V praxi však bývá rozdělení položek do pomocných souborů mnohem vyrovnanější. Jednotlivé pomocné soubory jsou setříděny samostatně a nakonec sloučeny do jednoho výstupního souboru.

První verze třídicího mechanismu, neúnosně pomalá (nevyužívala ani rozdělení položek do pomocných souborů), fungovala tak, že první položka ve vstupním souboru byla z tohoto souboru vyjmuta, uložena do paměti a postupně

porovnávána s ostatními položkami v tomto souboru. Byla-li položka v paměti z hlediska třídění shledána větší než ta, s níž byla porovnávána, byly obě položky prohozeny. Po projití celého vstupního souboru byl obsah paměti zapsán na konec výstupního souboru setříděných položek. To se opakovalo, než byl vstupní soubor zcela vyprázdněn. Vstupní soubor čítající n položek bylo tedy třeba projít $(n-1)$krát a celkově bylo provedeno $(n^2-n)/2$ porovnání položek.

Druhá verze byla chytřejší. Položka načtená do paměti se neporovnávala s netříděnými položkami vstupního souboru, nýbrž se setříděnými položkami výstupního souboru. Tím se počet porovnání položek stal závislý nejen na celkovém počtu položek, ale i na jejich původním pořadí; tento počet porovnání se pohyboval v rozmezí od $n-1$ do $(n^2-n)/2$, přičemž druhá krajní hodnota odpovídala setřídění vstupního souboru s n již abecedně seřazenými položkami, kdežto první krajní hodnota odpovídala setřídění vstupního souboru s n pozpátku seřazenými položkami.

Protože operace zápisu a čtení do/ze souboru jsou obecně pomalé, začal jsem experimentovat s tříděním pouze v paměti TEXu. Tato „bufferová“ metoda třídění byla opravdu rychlejší než původní „souborová“ metoda, ale pouze do určitého počtu položek; za touto hranicí se rozdíl rychlosti obou metod zvětšoval ve prospěch metody „souborové“, neboť TEX zpracovává dlouhé řetězce pomalu. V TEXu je totiž jistý exponenciální vztah mezi délkou řetězce a dobou jeho zpracování.

Aby bylo dosaženo co možná nejvyšší rychlosti třídění, výsledná verze třídicího mechanismu je kombinací obou dvou třídicích metod („bufferové“ a „souborové“). Nejdříve se v paměti TEXu setřídí skupina dosud nesetříděných položek, jejichž počet je dán registrem `\bufsize`; potom se tato skupina porovná s již setříděným souborem. A tak pořád dokolečka. Registr `\bufsize` je přednastaven na hodnotu 16.

Použití makra sort

Abecední řazení pomocí makra sort je velice snadné; stačí TEXem zpracovat soubor obsahující následující příkazy:

`\input sort`
`\sort` {*třídicí tabulka*} {*vstupní soubor*} {*výstupní soubor*}
`\bye`

Uživatel může experimentovat s nastavením registru `\bufsize`, který jistým způsobem ovlivňuje rychlost třídění; minimální hodnota je 2, maximální hodnota je omezena dostupnou pamětí TEXu. Příkazu `\sort` je možno v jednom dokumentu užít vícekrát.

Vstupní soubor se skládá z libovolného počtu příkazů

`\abc{`*položka*`}{`*text položky*`}`

Třídění se provádí podle parametru *položka*. Je-li v seznamu více příkazů `\abc` se stejnou *položkou*, ve výstupním souboru zůstane jen jeden; parametry *text položky* budou sloučeny a odděleny oddělovačem `\sep`. Každá *položka* může být rozdělena příkazy `\endoffield` do několika polí, což umožňuje implementovat rejstříky s podpoložkami, podpodpoložkami, podpodpodpoložkami atd. (tj. makeindexovský příkaz vykřičník). V *položce* se může vyskytnou příkaz `\endofentry`; v tom případě bude položka tříděna jen podle textu předcházejícího tomuto příkazu, ostatní text bude ignorován; to umožňuje implementovat makeindexovský příkaz zavináč. Všechny řídicí sekvence s výjimkou `\endofentry` a `\endoffield` jsou při abecedním řazení ignorovány.

Pravidla abecedního řazení jsou definována ve třídicí tabulce, která je uložena v samostatném souboru. Komentovaný příklad třídicí tabulky, vhodné pro češtinu, nalezneme v souboru `stab-cz.tex`. Třídicí tabulka obsahuje příkazy `\sorttab` a `\sorttabx`, které jednotlivým znakům přiřazují číslo abecedního pořádku. Příkaz `\sorttab` přiřazuje číslo pro každý průchod, zatímco příkaz `\sorttabx` přiřazuje číslo jen pro čtvrtý průchod. Řídicí sekvence s identifikátorem `p`*číslo průchodu*`:`*znak* se za pomoci příkazu `\chardef` stane číselnou konstantou. Znaky, jimž nebylo přiřazeno žádné číslo (tj. řídicí sekvence s identifikátorem `p`*číslo průchodu*`:`*znak* má význam příkazu `\relax`), jsou při třídění ignorovány. Všechny znaky uvedené společně jako parametr makra `\sorttab` mají pro první průchod přiděleno stejné číslo. Všechny znaky uvedené společně v jedné skupině, v níž jsou vzájemně oddělené čárkou, mají pro druhý průchod přiděleno stejné číslo; jednotlivé skupiny se od sebe oddělují středníkem. Pro třetí a čtvrtý průchod mají všechny znaky, které se vyskytují v parametru makra `\sorttab`, přidělena různá čísla. Například `\sorttab{e,E;é,É;ě,Ě}` znamená, že

v 1. průchodu e = E = é = É = ě = Ě,
ve 2. průchodu e = E < é = É < ě = Ě,
ve 3. průchodu e < E < é < É < ě < Ě,
ve 4. průchodu e < E < é < É < ě < Ě.

Znaky uvedené jako parametr makra `\sorttabx` mají přiděleno číslo jen pro čtvrtý průchod; vzájemně jsou odděleny čárkou. Za jeden znak je výjimečně možno pokládat skupiny písmen ch, Ch, CH, cH. Čárka a středník se do třídicí tabulky vkládají pomocí zástupných řídicích sekvencí `\comma` a `\semicolon`.

Cizí jazyky

Domnívám se, že při vytvoření vhodných třídicích tabulek bude čtyřprůchodový třídicí algoritmus makra sort dostatečný pro většinu evropských jazyků, včetně polytonické řečtiny. Čeština má tu zvláštnost, že při abecedním řazení pokládá

skupinu znaků ch za jediné písmeno. V jiných jazycích se vyskytuje opačný jev: jeden znak odpovídá dvěma písmenům, např. ß v němčině nebo œ ve francouzštině. To se dá lehce vyřešit prostřednictvím speciálních znaků na úrovni zapisovacích a čtecích maker: Například zařídíme, aby se do vstupního souboru pro rejstřík přepsalo každé ostré es jako dvojice znaků `^^19^^19`, do třídicí tabulky dáme řádek `\sorttab{s,^^19,S}` a při sazbě výstupního souboru zajistíme, aby dvojice znaků `^^19^^19` vytiskla jedno ostré es: `\catcode‘^^19=13 \def^^19^^19{\ss}`. Pak budeme mít hezky setříděno: Masse < Maße < Mast. Jisté starosti nám udělá také to, že v různých jazycích se stavějí různě k otázce víceslovných položek. Má být (Petr a Jan) < (Petra a Jana), nebo (Petra a Jana) < (Petr a Jan)? Ignorování mezer (např. ve všech průchodech kromě posledního) se dá snadno zařídit ve třídicí tabulce. Nastavení třídění nad jednotlivými slovy versus nad celou položkou v různých průchodech by však vyžadovalo zásah do maker `\firstpass`, `\secondpass`, `\thirdpass` a `\fourthpass`. Tolik o perspektivě využití makra pro cizojazyčnou sazbu.

Tvorba rejstříku

Chceme-li vytvářet rejstřík, musíme pomocí příkazu `\input index` načíst soubor maker `index.tex`; spolu s tímto souborem bude automaticky načten i soubor `sort.tex`. Pak pomocí příkazu `\openindex` otevřeme soubor `\jobname.idx`, do nějž budou v průběhu zpracování dokumentu ukládány podklady pro vytvoření rejstříku. Na konci dokumentu tento soubor zavřeme pomocí příkazu `\closeindex`. Stránka obsahující poslední odkaz do rejstříku musí být uložena do souboru `\jobname.dvi` dříve, než přijde ke slovu příkaz `\closeindex`; proto před tímto příkazem napíšeme něco jako `\vfill\superject`. Poté co je soubor `\jobname.idx` uzavřen, můžeme pomocí příkazu `\sortindex` rejstřík setřídit (výsledek této činnosti bude uložen do souboru `\jobname.ind`) a pak už nám jen zbývá použít příkaz `\printindex` k vytištění rejstříku. Nečekejme však, že makro `\printindex` vysází nadpis „Rejstřík“, zahájí víceslopcovou sazbu, změní velikost písma atd.; to jsou věci, které si uživatel musí podle konkrétního typografického požadavku zařídit sám.

Tvorbu rejstříku lze ovlivnit prostřednictvím několika parametrů. Pokud ještě před načtením souboru `index.tex` přiřadíme řídicí sekvenci `\makeindex` pomocí `\chardef` hodnotu 0 (`\chardef\makeindex=0`), rejstřík vytvořen nebude; této vlastnosti s prospěchem využijeme při vzniku dokumentu nebo při jeho dolaďování, kdy nám o sestavení rejstříku vůbec nejde. Pokud řídicí sekvenci `\makeindex` přiřadíme hodnotu 1, do souboru `\jobname.idx` budou uloženy podklady pro tvorbu rejstříku, ale rejstřík nebude setříděn ani vytištěn, neboť příkazy `\sortindex` a `\printindex` neprovedou příslušnou činnost. K úplnému vytvoření rejstříku vede hodnota 2; tu však řídicí sekvenci `\makeindex` přiřazovat nemusíme, neboť se jedná o implicitní nastavení.

Pro základní vzhled rejstříku jsou připraveny tři styly, označené čísly 0, 1 a 2. Styl 2 je vhodný pro rejstřík s maximálně dvěma úrovněmi hesel (tj. heslo a podheslo). Výběr stylu se provádí ještě před načtením souboru `index.tex` tím, že pomocí `\chardef` přiřadíme řídicí sekvenci `\indexstyle` příslušné číslo. Pokud tak neučiníme, použije se implicitně styl 0. Úpravou některých maker si uživatel může vytvořit svůj vlastní styl.

Jméno souboru s třídicí tabulkou, která se má použít pro abecední řazení rejstříku, se uvádí v těle definice makra `\stabfile`. Implicitně je definováno `\def\stabfile{stab-cz}`; to znamená, že se použije třídicí tabulka vhodná pro vytváření českých rejstříků.

V těle definice makra `\itabfile` se uvádí jméno souboru s inverzní tabulkou. Jak název napovídá, inverzní tabulka je, zhruba řečeno, opakem třídicí tabulky (v třídicí tabulce jsou znakům přiřazena čísla, zatímco v inverzní tabulce jsou číslům přiřazeny znaky) a využívá se jí při sazbě záhlaví skupiny hesel začínajících stejným písmenem. Zvolená inverzní tabulka by měla korespondovat se zvolenou třídicí tabulkou. Implicitně je definováno `\def\itabfile{itab-cz}`.

Text dokumentu je umístěn mezi příkazy `\openindex` a `\closeindex`. V textu můžeme pomocí různých příkazů vyznačit místa, na něž se má v rejstříku odkazovat.

Chceme-li do rejstříku zanést odkaz na jedno slovo, zapíšeme před to slovo příkaz `\idx` a do jeho parametru uvedeme heslo, pod nímž má být daný odkaz v rejstříku zařazen; například:

```
Zvláště \idx{Plutarch}Plutarch se stal mojí oblíbenou četbou.
```

Často ale nechceme, aby čísla stránek v rejstříku odkazovala jen na výskyt pouhého jednoho slova v textu, nýbrž na pasáž textu, kde se pojednává o pojmu vyjádřeném příslušným rejstříkovým heslem. V tom případě místo užití příkazu `\idx` uzavřeme mezi dvojici příkazů `\idxb` a `\idxe` danou pasáž textu; například:

```
Mrhání časem je velké zlo. Jiná, ještě horší zla provázejí vědy
a~umění. \idxb{přepych}Takový je přepych, zrozený jako ony ze
zahálky a~marnivosti lidí. Přepych jde zřídka bez věd a~umění
a~ony nikdy nejdou bez něj.\idxe{přepych}
```

Někdy chceme v rejstříku zdůraznit některé číslo stránky (např. tučným písmem), abychom čtenáře upozornili, že příslušná pasáž či příslušný výskyt daného slova je důležitější než ostatní odkazy. Pro vytvoření rejstříkového odkazu se zvýrazněným číslem stránky máme k dispozici následující příkazy:

`\idxit` — odkaz na jedno slovo, kurzivně;
`\idxbit` — začátek odkazu na pasáž textu, kurzivně;
`\idxeit` — konec odkazu na pasáž textu, kurzivně;
`\idxbf` — odkaz na jedno slovo, tučně;
`\idxbbf` — začátek odkazu na pasáž textu, tučně;

`\idxebf` — konec odkazu na pasáž textu, tučně;
`\idxul` — odkaz na jedno slovo, podtrženě;
`\idxbul` — začátek odkazu na pasáž textu, podtrženě;
`\idxeul` — konec odkazu na pasáž textu, podtrženě.

Odkaz na jiné rejstříkové heslo („viz“, „viz též“) se do zdrojového textu zapisuje pomocí příkazu `\idxsee`; například:

```
\idxsee{Istanbul}{Konstantinopol}
```

Pro oddělení hesel, podhesel, podpodhesel atd. se v parametru výše uvedených příkazů užívá vykřičníku; například:

```
\idx{jazyk!indoevropské jazyky!slovanské jazyky!čeština}
```

Pokud si přejeme řadit některé heslo jinak, než by odpovídalo jeho psané podobě, můžeme použít zavináč k oddělení podoby hesla, která se má vytisknout, od podoby, podle níž se má řadit; například:

```
\idx{Ludvík 08@Ludvík~VIII.}
\idx{Ludvík 09@Ludvík~IX.}
\idx{Ludvík 10@Ludvík~X.}
```

V jednom parametru příkazu pro vytvoření rejstříkového odkazu se může vyskytnout zavináč i vykřičník; například:

```
\idx{král!Ludvík 14@král!Ludvík~XIV.}
```

Všechny výše uvedené příkazy pro tvorbu rejstříkových odkazů, vyskytnou-li se ve vertikálním módu, způsobí přechod do odstavcového módu. To je důležité pro to, aby do rejstříku bylo zaneseno správné číslo strany, když se těchto příkazů užije na samém začátku odstavce. Potřebujeme-li výjimečně potlačit přechod do odstavcového módu, můžeme využít například následujícího triku: `\hbox{\idx{`*heslo*`}}`.

Pro zpracování čísel stran platí tyto čtyři zásady:

1. Na uživatelem zadaný rozsah stran (např. pomocí dvojice příkazů `\idxb` a `\idxe`) se pohlíží, jako by do rejstříku byly zaneseny jednotlivé strany z tohoto rozsahu.

2. Každé číslo strany se v rejstříku vyskytne pouze jednou; způsoby zvýraznění jednotlivých čísel se kumulují.

3. Posloupnost dvou či více těsně za sebou jdoucích čísel, která jsou popřípadě tímtéž způsobem zvýrazněna, se převádí na rozsah (např. posloupnost čísel 1, 2, 3, 4, 5, 6, *7*, *8*, *9* bude zpracována takto: 1–3, 4–6, *7*, *8–9*).

4. Záporná čísla stran (tj. strany předmluvy), jejichž pořadí musí být sestupné, se zpracovávají stejně jako kladná čísla s tím rozdílem, že budou vytištěna římskými číslicemi (malými písmeny).

Pokud v jednom rejstříku zvýrazňujeme čísla odkazovaných stran kurzivně i tučně, musíme definovat řídicí sekvenci `\bi` jako přepínač do tučné kurzivy pro případ, že by některé číslo mělo být zvýrazněno oběma způsoby současně.

Pro sazbu rozsahové pomlčky můžeme využít makra `\až`, popisovaného v článku *Hrstka tipů pro TEXovskou sazbu* [1]. V tom případě musíme náležitým způsobem předefinovat příkaz `\spandash`.

Při zápisu do souboru `\jobname.idx` je zapotřebí změnit kategorii aktivních znaků na neaktivní, tak aby neexpandovaly nežádoucím způsobem; to může uživatel zařídit v definici makra `\idxdefs`. Podobnou úlohu při načítání třídicí a inverzní tabulky plní makro `\tabdefs`; zde je však nutné definovat návrat k původnímu nastavení prostřednictvím makra `\canceltabdefs`.

Bibliografie

[1] Březina, Petr: Hrstka tipů pro TEXovskou sazbu. Zpravodaj Československého sdružení uživatelů TEXu, 16/2006, 1, str. 10—25.

[2] Olšák, Petr: Program csr (Czech SoRt) — abecední řazení podle normy. Zpravodaj Československého sdružení uživatelů TEXu, 94, 3, str. 126—139.

[3] Olšák, Petr: Typografický systém TEX. 2. vyd., Brno, Konvoj, 2000.

[4] Winograd, Terry — Paxton, Bill: An indexing facility for TEX. TUGboat, 1, 1, Appendix A, str. 1—12.

Summary: Alphabetical sorting and creation of indexes exclusively by means of TEX language

The article presents the macro package “index” for creation of indexes. An important part of this package is the TEX macro “sort” for alphabetical sorting. It uses a four-pass sorting algorithme which can be accommodated to different languages through a sorting table. Its memory requirements are independent of the length of the list of entries to be sorted. The treatment of page numbers in index entries is sophisticated. The macro package “index” is available including French documentation on author’s home page http://www.volny.cz/petr-brezina/.

Proměnné záhlaví a zápatí

Pavel Stříž

Tento článek se zabývá sazbou záhlaví a zápatí. Popisuje základní možnosti a způsoby, jak je vysázet. K tomu využívá ve většině ukázek standardního balíčku `fancyhdr`. Ukazuje možnosti přípravy proměnných prvků, které jsou obvykle závislé na aktuální stránce. V případě, že jsou proměnné prvky navíc navzájem provázané, článek zmiňuje metodu, která část nebo celý TEXový dokument vygeneruje za pomoci `PHP` a `MySQL`.

Závěrečná ukázka, její zdrojový kód a všechny očíslované zdrojové partie jsou nezávisle publikovány na serveru `http://bulletin.cstug.cz/`.

Klíčová slova: Sazba a proměnlivost v záhlaví a zápatí, balíček `fancyhdr`, programování v `PHP` a `MySQL`.

This article deals with the typesetting of headings and footings. It describes basic opportunities and ways how to typeset them. It uses the standard package `fancyhdr` nearly on all examples. It shows the ways how to prepare variable objects which are usually page dependent. In case variable objects are dependent itself in addition, the article introduces a method which generates part or whole TEX document using `PHP` and `MySQL` tools.

Final example, its source code and all numbered source codes are independently published on `http://bulletin.cstug.cz/`.

Key words: Headings and footings typesetting and its variability, `fancyhdr` package, `PHP` and `MySQL` programming.

Предметом настоящей статьи является (типографский) набор колонтитулов. Описаны основные варианты набора (верхних и нижних) колонтитулов. Почти все примеры в тексте пользуются стандартным пакетом `fancyhdr`. Показаны способы определения изменяемых объектов обычно зависящие от содержания страницы. В случае зависимости изменяемых объектов от других изменяемых объектов, настоящая статья предоставляет методы генерирования частей или полного TEX документа с помощью средств `PHP` и `MySQL`.

Заключительный пример, его исходный код и все остальные нумерованые исходные коды опубликованы на `www.cstug.cz`.

Ключевые слова: Набор колонтитулов и его изменяемость, `fancyhdr` пакет, программирование на `PHP` и `MySQL`.

1. Seznámení s problémem

Se zajištěním proměnného záhlaví a zápatí, tedy s tímto problémem, jsem se setkal při výpomoci nad skripty psanými v programu Microsoft Word®, kde byl požadavek (spíše přání) mít odděleně vysázeno na každé straně v zápatí jiný citát a jeho autora. To šlo pomocí nastavení pevných oddílů v závěru publikace zrealizovat, ale při požadavku na změnu pořadí, přidání, vypuštění citátů, nebo při zásazích do textu by nastal ruční blázinec.

Není proto úplně na škodu nahlédnout na nástroje TEXu a mít vhodný nástroj po ruce na podobné situace do budoucna.[1]

2. Sazba záhlaví a zápatí

Nejčastěji pracujeme s `\markboth`, který je vhodný i pro zrcadlovou sazbu, a za pomoci `\def\ps@` a příkazů `\def\@oddhead`, `\def\@evenhead`, `\def\@oddfoot` a `\def\@evenfoot`. Lze použít i balíček `fancyhdr` (nebo modifikovanou alternativu `nccfancyhdr`), se kterým můžeme bez problémů rozmístit objekty do dvanácti směrů při zrcadlové sazbě. Pokud bychom potřebovali rozmístit ještě více objektů, lze to zařídit dle našich vlastních představ přes příkaz `\makebox`.

U knih využíváme různě naformátované sekce a podsekce s číslováním nebo bez něj [4, str. 113 – 115], a to při zrcadlové sazbě. U zrcadlové sazby nastavujeme tzv. zrcadlo sazby. To má plně využít potenciálu oboustranného tisku a zajistit maximální čitelnost dokumentu. Pojem zrcadlová sazba může vyznít i trochu jinak (viz grafická vsuvka, str. 45).

U slovníků se lze setkat v záhlaví vlevo a vpravo s prvním a posledním pojmem vyskytujícím se na dané straně [4, str. 115]. Typograficky obdobnou situaci řešil i pan Olšák na přednášce v Brně v listopadu roku 2003 při sazbě inzertní přílohy časopisu Dotek [2].

S proměnným charakterem v záhlaví a zápatí se lze setkat v podsekci *A movie* [4, str. 121]. Zde se předpokládá již předpřipravená sada obrázků a při odstranění jednoho z nich se sada obrázků za tímto vyřazeným obrázkem již nepřepočítává.

Pokusil jsem se s kolegou nastínit proměnné záhlaví a zápatí již dříve [5].

2.1. Styly `headings` a `myheadings`

Začněme běžně zažitou ukázkou:

```
1 \documentclass[twoside]{article}
2 \pagestyle{headings}
```

[1] Motto *To i my jsme zamlada psávali jen tím uživatelsky přívětivým stylem* `empty` ...

Mé poděkování patří pánům Jozefu Říhovi, Radku Vicherkovi, Miroslavu Červenkovi a Haně a Martinu Střížovým za pečlivé pročtení rukopisu. Poděkování patří i panu Igoru Novozámskému za vytvoření zakázkové grafické vsuvky.

```
\begin{document}
%...                                Zde patří text těla dokumentu.
\end{document} \bye
```

Tímto si lze zajistit vhodné záhlaví a zápatí, kterým téměř nikdo nepohrdne. Pokud si čtenář není u editace jistý zdrojovým kódem, např. u `article.cls`, může využít o něco méně elegantní metodu za pomoci `myheadings`:

```
\documentclass[twoside]{article}
\pagestyle{myheadinds}
\begin{document}
\section[text_1]{text_2}\markboth{text_2}{text_5}
%...                                                    Běžný text.
\section[text_3]{text_4}\markboth{text_4}{text_6}
%...                                                    Běžný text.
\end{document} \bye
```

Bez hlubších znalostí pak lze při každé změně např. `\section` a `\subsection` měnit texty i v parametrech `\markboth`. Tím se můžeme dostat náročnější cestou k první ukázce. Vhodné je to však ve chvíli, kdy nechceme použít automatizovaných prvků. Anebo přesně obráceně, kdy chceme vysázet, např. do záhlaví, dohromady část, sekci i podsekci. Je to typografická zajímavost a čtenář se s tím setká jen málokdy. Minimálně to upoutá pozornost (viz podsekce 2.6).

2.2. Styl `myfootings`

Neodvracejme však naši tvář od automatizovaných prvků. Nejednomu začínajícímu TEXistovi se může zdát, že zápatí je příliš nevytížené a chce si záhlaví se zápatím u `myheadings` prohodit. Pak stačí jen vyhledat `article.cls`, konkrétně definování `myheadings` a `myfootings` si připravit takto:

```
\def\ps@myfootings{%
    \let\@oddhead\@empty\let\@evenhead\@empty
    \def\@evenfoot{\thepage\hfil\slshape\leftmark}%
    \def\@oddfoot{{\slshape\rightmark}\hfil\thepage}%
    \let\@mkboth\@gobbletwo
    \let\sectionmark\@gobble
    \let\subsectionmark\@gobble }
```

Je možné si styl přidat do `article.cls`, do vlastního souboru, nebo si tyto řádky nechat v preambuli dokumentu.

Není úplně na škodu se také podívat v `article.cls` na přípravu `headings`, protože někdy stačí upravit jen málo, abychom byli se svým záhlavím a zápatím spokojeni. Nechť se začínající TEXista nenechá odradit příkazem `\if@twoside`, protože to je klíč k různému záhlaví a zápatí u běžné a zrcadlové sazby.

Pokud chceme mít náš dokument z pohledu záhlaví a zápatí prázdný, použijeme `\pagestyle{empty}`. Pokud bychom chtěli zvolit konkrétní styl, byť právě na jedné straně (nyní stylem `empty`), použijeme `\thispagestyle{empty}`.

2.3. Balíček `fancyhdr`

U tříprvkových záhlaví a zápatí, kde jsou levé a pravé položky proměnných prvků nesymetrické, je střed (při použití `\hfill`) vychýlen a s tím nejsme spokojeni jako doposud. Buď začneme s přeměřováním objektů nebo využijeme balíčku `fancyhdr`, který si zajistíme v preambuli pomocí:

```
21 \usepackage{fancyhdr}
22 \pagestyle{fancy}
```

Je dobré si pročíst dokumentaci a zdrojový soubor k tomuto balíčku v podobě `fancyhdr.sty` (nebo `nccfancyhdr.sty`). Lze totiž načerpat řadu podnětů ke zpracování záhlaví a zápatí. Balík `fancyhdr` nám umožňuje při zrcadlové sazbě využít 12 míst a pomocí několika příkazů rozmístit prvky do záhlaví a zápatí našeho dokumentu (viz nákres níže).

```
23 \fancyhf{}
24 \renewcommand{\headrulewidth}{1.2pt}
25 \renewcommand{\footrulewidth}{0.8pt}
26 \fancyhead[LE,RO]{\slshape \rightmark}
27 \fancyhead[LO,RE]{\slshape \leftmark}
28 \fancyfoot[C]{\thepage}
```

Pomocí řádku 23 si vyčistíme naše záhlaví a zápatí, řádky 24 a 25 si nastavíme linky a v dalších třech řádcích říkáme co chceme mít zobrazeno. Jsme tam, kde jsme začali – u volby `headings`. Ovšem s tou výhodou, že můžeme velmi nenásilně operovat na celé dvoustraně, a to jak v záhlaví, tak v zápatí. Velmi rychle se můžeme přesvědčit i o nenásilné práci na více řádcích pomocí zalamování `\\`. Nastavení pracovních rozměrů záhlaví a zápatí můžeme nastudovat např. z [1, str. 110 – 112].

	přes `\hfill`	vlevo (left)	uprostřed (center)	vpravo (right)
	přes `\makebox`	vlevo (left)	uprostřed (center)	vpravo (right)
Lichá strana (odd)	`\fancyhead`	LO	CO	RO
		Tělo dokumentu		
	`\fancyfoot`	LO	CO	RO
			⋮	
Sudá strana (even)	`\fancyhead`	LE	CE	RE
		Tělo dokumentu		
	`\fancyfoot`	LE	CE	RE

Pokud bychom chtěli nastavit okraje záhlaví a zápatí delší než okraje těla dokumentu, nahlédneme na [4, str. 12 a 13].

2.4. Speciální rozmístění

Pokud chceme rozmístit objekty velmi přesně, a to nezávisle na předchozím sázeném textu (viz pokus o vycentrování v záhlaví nákresu výše), můžeme v záhlaví a zápatí při využití balíčku `fancyhdr` zkusit tuto alternativu:

```
29 \fancyhf{}
30 \newcommand{\vboxu}[1]{\makebox[0pt][l]{#1}}
31 \fancyhead[l]{\vboxu{\thepage\ text poměrně dlouhý,
32    mnohem delší než ten v zápatí}\hspace{9cm}text}
33 \fancyfoot[l]{\vboxu{\thepage\ krátký text}\hspace{9cm}text}
```

Tímto obratem na řádcích 29 – 33 jsme získali možnost přesně umístit text nebo grafiku tak, aby posuny nebyly ovlivněny délkou sazby textu nebo grafiky na tom samém řádku. Takto můžeme dle svého uvážení rozmístit typografické prvky. Pokud zvolíme vhodnou proměnnou místo devíti centimetrů, lze rozmístění řídit poměrně rozumně.

Na tomto příkladě je již vidět, že ačkoliv dokážeme automaticky rozmístit prvky, zatím nemáme nástroj na editaci jednotlivých položek dostatečně flexibilní vůči každé straně dokumentu. Doposud jsme jen zvažovali jisté změny v záhlaví a zápatí na úrovni `\section`, `\subsection` a dalších dokument zpřehledňujících prvcích.

2.5. Změna stylu

Podobně jako `myfootings` si můžeme připravit celou řadu zajímavých stylů a v dokumentu se mezi nimi přepínat. Nejčastěji potřebujeme v publikaci tři až čtyři styly. Na úvodní strany knihy se dává jeden styl (čísla stránek římskými číslicemi). Dokument je sázen v dalším stylu (nejčastěji `headings` nebo `myheadings`). U příloh a dodatků se setkáváme s jednodušším stylem bez čísel stran. U první strany a občas i u rozsáhlejších ukázek dáváme styl `empty`.

Přepínání realizujeme v textu následujícím způsobem:

```
34 \pagestyle{empty}                        %Zavedení prvního stylu.
35 %...                                              Text dokumentu.
36 \pagestyle{headings}                     %Zavedení druhého stylu.
37 %...                                              Text dokumentu.
38 \thispagestyle{myfootings} %Třetí styl. Určen pro jednu stranu.
39 %...               Text dokumentu. Pokračuje se stylem headings.
40 \pagestyle{empty}                     %Zpátky na první styl. Atd.
```

Pokud si nechceme definovat styly, můžeme si nadefinovat přímo příkazy:

```
\def\stylA{\fancyhf{}}
\def\stylB{\fancyhf{} \fancyhead[L]{Konference}
   \fancyhead[R]{\thepage} \fancyfoot[L]{Místo a čas}}
\def\stylC{\thispagestyle{fancy} \fancyhf{}
   \fancyhead[C]{\partname ~\thepart ~--- \leftmark} }
```

V textu se mezi styly balíčku `fancyhdr` pohybujeme podobně:

```
\stylA                                          %Zavedení stylu A.
%...                                               Text dokumentu.
\stylB                                          %Zavedení stylu B.
%...                                               Text dokumentu.
\stylC                 %Zavedení stylu C. Určen pro jednu stranu.
%...           Text na jedné straně a pak se pokračuje stylem B.
\stylA                                     %Zpátky na styl A. Atd.
```

2.6. Řízené změny stylu

Přepínat se jako v předchozí podsekci není po určité době a častých změnách vždy vhodná možnost. Ukažme si způsob jak využít automatických čítačů na třech úrovních nadpisů. S každou ze zmíněných čtyř variant (vysázeno jako bloky) lze pracovat relativně samostatně.

Nejjednodušší variantou je zobrazit aktuální nadpis. Na aktuální straně se vysází vždy ten poslední nadpis. Rozšířili jsme tedy `\part` na `\urovenA`, `\section` na `\urovenB` a `\subsection` na `\urovenC`.

```
\def\urovenA#1{\part{#1}\fancyhf{}
   \fancyhead[C]{\partname\ \thepart\ \emph{#1}} }
\def\urovenB#1{\section{#1}\fancyhf{}
   \fancyhead[C]{Sekce \thesection\ \emph{#1}} }
\def\urovenC#1{\subsection{#1}\fancyhf{}
   \fancyhead[C]{Podsekce \thesubsection\ \emph{#1}} }
```

Strana 1	záhlaví	Podsekce 1.1 *Historické poznámky*
		Část I – Umělé neuronové sítě 1 Kohonenovy sítě 1.1 Historické poznámky
	zápatí	PRÁZDNÉ
		⋮
Strana n	záhlaví	Podsekce 3.7 *Citované zdroje*
		3.5 Průmyslové využití 3.6 Příklady a cvičení 3.7 Citované zdroje

	zápatí	PRÁZDNÉ

Druhou variantou je sazba posledního nadpisu na stránce, ale v úplné formě. Vysází se název úrovně nadpisu, její hodnota a její název. U úrovně B se zobrazí i úroveň A a u úrovně C se zobrazí A i B. V zápatí se zobrazí zkrácená varianta jen za použití arabských číslic. Zvolíme tuto cestu:

```
\def\urovenA#1{\part{#1}\def\textA{\emph{#1}}\fancyhf{}
   \fancyhead[C]{\partname\ \thepart\ \textA}
   \fancyfoot[C]{\thepart} }
\def\urovenB#1{\section{#1}\def\textB{\emph{#1}}\fancyhf{}
   \fancyhead[C]{\partname\ \thepart\ \textA\ ---\ Sekce\ %
   \thesection\ \textB}
   \fancyfoot[C]{\thepart\,--\,\thesection} }
\def\urovenC#1{\subsection{#1}\def\textC{\emph{#1}}\fancyhf{}
   \fancyhead[C]{\partname\ \thepart\ \textA\ ---\ Sekce\ %
   \thesection\ \textB\ ---\ Podsekce\ \thesubsection\ \textC}
  \fancyfoot[C]{\thepart\,--\,\thesection\,--\,\thesubsection}}
```

Strana 1	záhlaví	Část I *JAVA* – Sekce 1 *Programátorské milníky* –
		– Podsekce 1.1 *Konec 20. století*
		Část I – JAVA
		1 Programátorské milníky
		1.1 Konec 20. století
	zápatí	I – 1 – 1.1
		⋮
Strana n	záhlaví	Část IX *Síťové protokoly* – Sekce 11 *Rodina TCP* –
		– Podsekce 11.1 *TCP/IP*
		Část IX – Síťové protokoly
		11 Rodina TCP
		11.1 TCP/IP
	zápatí	IX – 11 – 11.1

Alternativně vůči první variantě zmíníme sazbu nadpisů s tím rozdílem, že je vysázen nadpis na zkoumané straně první:

```
\newcount\stopka\newcount\posledni\posledni=0
\def\textA{}\def\textB{}\def\textC{}
\newcounter{hodnotaA}\newcounter{hodnotaB}\newcounter{hodnotaC}
\def\urovenA#1{\part{#1}
```

```
   \ifnum\posledni=\thepage\else
   \def\textA{\emph{#1}}\setcounter{hodnotaA}{\value{part}}
   \fancyhf{}\fancyhead[C]{\partname\ \Roman{hodnotaA}\ \textA}
   \posledni=\value{page}\fi }
\def\urovenB#1{\section{#1}
   \ifnum\posledni=\thepage\else
   \def\textB{\emph{#1}}\setcounter{hodnotaB}{\value{section}}
   \fancyhf{}\fancyhead[C]{Sekce\ \thehodnotaB\ \textB}%
   \posledni=\value{page}\fi }
\def\urovenC#1{\subsection{#1}
   \ifnum\posledni=\thepage\else\def\textC{\emph{#1}}
   \setcounter{hodnotaC}{\value{subsection}}
   \fancyhf{}\fancyhead[C]{Podsekce\ \thehodnotaC\ \textC}%
   \posledni=\value{page}\fi }
```

Strana 1	záhlaví	Část I *Umělé neuronové sítě*
		Část I – Umělé neuronové sítě 1 Kohonenovy sítě 1.1 Historické poznámky
	zápatí	Prázdné
		⋮
Strana n	záhlaví	Podsekce 3.5 *Průmyslové využití*
		3.5 Průmyslové využití 3.6 Příklady a cvičení 3.7 Citované zdroje
	zápatí	Prázdné

Čtvrtou a poslední variantou řízené změny zmíníme rozšířenou modifikaci druhé varianty. V záhlaví je úplný výpis nadpisu na prvním místě aktuální strany a v zápatí je zkrácený zápis. Opět pracujeme se třemi úrovněmi a zavádíme několik pracovních proměnných.

```
\newcount\stopka\newcount\posledni\posledni=0
\def\textA{}\def\textB{}\def\textC{}
\newcounter{hodnotaA}\newcounter{hodnotaB}\newcounter{hodnotaC}
\def\urovenA#1{\part{#1}\def\docasneA{\emph{#1}}
   \ifnum\posledni=\thepage\else
   \def\textA{\emph{#1}}\setcounter{hodnotaA}{\value{part}}
   \fancyhf{}\fancyhead[C]{\partname\ \Roman{hodnotaA}\ \textA}
   \fancyfoot[C]{\Roman{hodnotaA}}
```

```
96      \posledni=\value{page}\fi }
97  \def\urovenB#1{\section{#1}\def\docasneB{\emph{#1}}
98      \def\textA{\docasneA}\setcounter{hodnotaA}{\value{part}}
99      \ifnum\posledni=\thepage\else
100     \def\textB{\emph{#1}}\setcounter{hodnotaB}{\value{section}}
101     \fancyhf{}\fancyhead[C]{\partname\ \Roman{hodnotaA}\ %
102     \textA\ ---\ Sekce\ \thehodnotaB\ \textB}%
103     \fancyfoot[C]{\Roman{hodnotaA}\,--\,\thehodnotaB}
104     \posledni=\value{page}\fi }
105 \def\urovenC#1{\subsection{#1}
106     \def\textA{\docasneA}\setcounter{hodnotaA}{\value{part}}
107     \def\textB{\docasneB}\setcounter{hodnotaB}{\value{section}}
108     \ifnum\posledni=\thepage\else\def\textC{\emph{#1}}
109     \setcounter{hodnotaC}{\value{subsection}}
110     \fancyhf{}\fancyhead[C]{\partname\ \Roman{hodnotaA}\ %
111     \textA\ ---\ Sekce\ \thehodnotaB\ \textB\ ---\ Podsekce\ %
112     \thehodnotaC\ \textC} \fancyfoot[C]{%
113     \Roman{hodnotaA}\,--\,\thehodnotaB\,--\,\thehodnotaC}
114     \posledni=\value{page}\fi }
```

Strana 1	záhlaví	Část I *JAVA*
		Část I – JAVA 1 Programátorské milníky 1.1 Konec 20. století
	zápatí	I
		⋮
Strana n	záhlaví	Část IX *Síťové protokoly*
		Část IX – Síťové protokoly 11 Rodina TCP 11.1 TCP/IP
	zápatí	IX

U třetí a čtvrté varianty jsme u pomocných čítačů nepoužili závislou proměnnou `section`, a tak je podsekce sázena jen jako prostý čítač, např. z „Podsekce 11.1 *TCP/IP*“ se vysází „Podsekce 1 *TCP/IP*“. Pokud chceme původní formát, jedna z možností je smazat řádky 112 a 113 a rozšířit je na tyto tři:

```
115     \thehodnotaB .\thehodnotaC\ \textC} \fancyfoot[C]{%
116     \Roman{hodnotaA}\,--\,\thehodnotaB\,--\,
117     \thehodnotaB .\thehodnotaC}
```

Používaná varianta je sazba nadpisů v záhlaví a zápatí, jejichž text se první (poslední) vyskytuje na aktuální straně. Tohle úspěšně vyřešil pan Olšák [2].

Lze zvolit alternativu pomocí ukládání začátků a konců nadpisů buď přes citování (s uloženým odkazem na stranu lze pracovat jako s číslem) nebo přes ukládání do dvou pracovních proměnných. Tyto metody považuji za nehotové a dovolím si je neuvádět. Ať laskavý čtenář promine, pokud právě tyto alternativně vyřešené partie v článku očekával.

3. Práce s \thepage

Proměnná `\thepage` vzniká automaticky po nastavení čítače `page`. Velmi často je třeba změnit jeho formát, a to zařídíme takto:

```
\renewcommand\thepage{\roman{page}}
```

Pokud se chceme dostat zpět k zápisu arabskými číslicemi, tak přes:

```
\renewcommand\thepage{\arabic{page}}
```

Některým TEXistům vyhovuje více alternativa `\pagenumbering{arabic}`. Vedle `arabic` a `roman` lze použít `Roman`, `alph` nebo `Alph`. Vlastní formát čísla strany lze připravit příkazy `\ifnum` nebo `\ifcase` (viz podsekce 3.1 a 3.2).

V další části textu budeme již pracovat s `\ifodd` a `\ifcase`. Toto je jedna z cest, jak si zajistit vhodný formát čítače `page`. Druhou cestou nám může být práce přímo s `\value{page}`, což je na formátu nezávislé.

3.1. Práce s \ifodd a \ifnum

Chceme-li vysázet text nebo vykreslit grafiku s ohledem na to, jsme-li na sudé nebo liché stránce, učiníme to takto:

```
Jsem na \ifodd\value{page}
liché \else
sudé \fi straně.
```

Výpis sudé/liché stránky lze přes novou proměnnou zařídit i pomocí příkazu `\ifnum` touto cestou:

```
\newcount\citatel \citatel=\thepage
\divide\citatel by 2 \multiply\citatel by 2
\ifnum\citatel=\value{page} Sudé, dvěma dělitelné \else
Liché, dvěma nedělitelné \fi číslo strany.
```

3.2. Práce s \ifcase

Pokud chceme do textu vypsat na které jsme straně, použijeme `\ifcase` takto:

```
127 \ifcase\value{page}
128 Jsem na nulté\or
129 Jsem na první\or
130 Jsem na druhé\else
131 Nejsem ani na nulté, první nebo druhé
132 \fi straně.
```

Pokud používáme formát `arabic`, můžeme si dovolit nahradit řádek 127 za `\ifcase\thepage`. Je dobré si uvědomit, že první záznam patří nule. Takový záznam můžeme přeskočit při změně řádku 128 na `\relax\or` nebo jen `\or`.

3.3. Posun v \ifcase

Uvedeme si jednoduchý příklad, v němž chceme slovně vypsat na které jsme straně, ale vůči kapitole, která je hluboko za nultou stranou. Pak je méně efektivní vypisovat sérii příkazů `\or` vedle sebe.

Můžeme použít učebnicového příkladu pana Olšáka [3, str. 375, `\ifcase`] za pomoci nové proměnné:

```
\newcount\tatostrana \newcount\odskok
\tatostrana=\value{page}
```

Tím si do proměnné `\tatostrana` uložíme začátek (např. kapitoly), a kdekoliv v textu pak můžeme operovat následujícím způsobem:

```
\odskok=\value{page} %Následují a/nebo předcházejí jiné strany.
\advance\odskok by -\tatostrana \advance\odskok by 1
\ifcase\odskok\or
První \or Druhá \or Třetí \else
Čtvrtá nebo vyšší \fi strana této kapitoly.
```

Můžeme však zasáhnout přímo do čítače `page`. Ten upravíme před a po `\ifcase`. Hodnota čítače mimo `\ifcase` zůstane stejná; dle svých zásahů však pracujeme s upravenou hodnotou tohoto čítače. Realizujeme to např. takto:

```
\addtocounter{page}{-6}
\ifcase\value{page}
Nultý \or První \or Druhý \or Třetí \else Jiný \fi záznam.
\addtocounter{page}{6}
```

3.4. Uvnitř záhlaví a zápatí

Nyní jsme se dostali do velice příjemné situace, kdy můžeme uvnitř záhlaví a zápatí použít `\ifcase`, a to přímo ve vztahu k čítači `page`. Obzvlášť přes balíček `fancyhdr` můžeme operovat i s jinými čítači velmi nenásilně.

Příklad. Chceme ve středu zápatí mít informaci, je-li strana menší než 100, přesně rovna 100 nebo je-li strana větší než 100.

Pomocí `\ifcase` bychom ke kýženému výsledku také dospěli, ale použijeme přímo `\ifnum`.

```
\fancyfoot[C]{\ifnum\value{page}<100 Jsem pod stranou sto.\else
\ifnum\value{page}=100 Jsem na straně sto. \else
Jsem na straně vyšší než sto. \fi \fi }
```

Pokud bychom všechny „100“ a „sto“ nahradili za proměnné, tak máme opět velmi flexibilní řešení. Úkol splněn.

Příklad. Chceme mít v záhlaví u vnitřního okraje dle čísla stránky v zrcadlové sazbě název měsíce.

Realizace může vypadat například takto:

```
\fancyhf{}
\fancyhead[LO,RE]{\ifcase\value{page}\or
Leden\or Únor\or Březen\or Duben\or Květen\or Červen\or
Červenec\or Srpen\or Září\or Říjen\or Listopad\or Prosinec
\else \relax \fi }
```

Velmi rychle lze změnit popisky na libovolný jiný jazyk. Úkol splněn.

Najdeme-li vhodnou metodu generování zdrojového kódu pro záhlaví a zápatí, budeme mít zajištěno jak automatické přečíslování, tak čistý TEXový dokument. K tomuto bodu se ještě vrátíme.

4. Hrátky s grafikou

U sazby grafiky nesmíme zapomenout uvést `\usepackage{graphicx}` v preambuli dokumentu. Osvědčila se mi práce v prostředí `picture`.

4.1. Sazba v oblasti záhlaví a zápatí

Uveďme si ukázku sazby textu a grafiky (vkládaný soubor `obr1` reprezentuje jednoduchou dekoraci). Horizontální linka nám v naší ukázce odděluje záhlaví od těla dokumentu.

```
\fancyhead[L]{\begin{picture}(0,0)
\put(0,0){\rotatebox[origin=r]{180}{
\includegraphics[height=0.75em, width=5em]{obr1}
text TEXT text \rotatebox[origin=r]{180}
{\includegraphics[height=0.75em, width=5em]{obr1}}
} } \end{picture} }
```

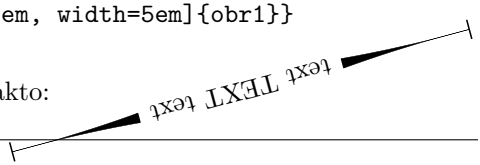
Příklad může vypadat například takto:

4.2. Sazba mimo oblast záhlaví a zápatí

I zde nám v příkladu horizontální linka odděluje záhlaví od těla dokumentu. Při změně parametrů na umístění u `\put` a změně natočení u `\rotatebox` jsme se octli mimo oblast záhlaví, i když to typograficky stále pod záhlaví patří.

```
\fancyhead[L]{\begin{picture}(0,0)
\put(170,-15){\rotatebox[origin=r]{195}{
\includegraphics[height=0.75em, width=5em]{obr1}
text TEXT text \rotatebox[origin=r]{180}
{\includegraphics[height=0.75em, width=5em]{obr1}}
} } \end{picture} }
```

Příklad může vypadat například takto:



Nyní tedy můžeme využít celé plochy papíru, především všech okrajů, a nenarušíme tím sazbu v těle dokumentu.

5. Jádro proměnného záhlaví a zápatí

Jeden z nápadů lze objevit v podsekci *A movie* [4, str. 121]. Někde si však obrázky musíme připravit a k tomu nemusíme použít přímo TEX. Podobně je to i s textem. Například u citátů, které si průběžně ukládáme do textového souboru, internetové databáze nebo např. do tabulkového procesoru v Open Source balíku OpenOffice.org. Převody mezi těmito třemi softwarovými součástmi nečiní většinou velké potíže.

5.1. Stránkově nezávislé prvky

Pokud se pohybujeme na úrovni stránkově nezávislých prvků, můžeme editovat přímo hlavní dokument nebo vhodněji z zpohledu přehlednosti editujeme zvláštní soubor, který si do hlavního dokumentu načteme. Načítáme ovšem i partie vztahující se k záhlaví a zápatí, abychom nevkládali soubor do prostředí.

V hlavním dokumentu načteme soubor `zahlavi.tex`, nejlépe v preambuli, pomocí `\input{zahlavi.tex}`. Soubor `zahlavi.tex` může vypadat takto:

```
\fancyhead[LO,RE]{\ifcase\value{page}\or
3,1\or
3,14\or
3,141\or
3,1415\or
3,14159\else
$\pi$ \fi }
```

```
\fancyhead[LE,RO]{\ifcase\value{page}\or
Kolega Hlaváč\or
Kolegyně Novotná\or
Kolegyně Tichá\or
Kolega Kafka\or
Kolega Havelka\else
--- \fi }
```

Je výhodné mít každou položku na jednom řádku – kvůli poměrně přehledné možnosti editace.

Ačkoliv máme všechny údaje v záhlaví, není mezi těmito údaji souvislost a můžeme každý proměnný prvek editovat zvlášť. Na tuto formu editace stačí editovat přímo tento soubor. Kdyby totiž tyto proměnné prvky byly závislé a chtěli bychom odebrat nebo přehodit údaje v prvním proměnném prvku, museli bychom ty samé kroky udělat i u druhého proměnného prvku. To při rozsáhlejších úpravách příjemné již není.

Přímá editace souboru `zahlavi.tex` je výhodná tam, kde jsou proměnné prvky stránkově nezávislé nebo je proměnný prvek jen jeden.

5.2. Stránkově závislé prvky

Za stránkově závislé prvky považujeme takové prvky, které při editaci prvního prvku dávají do souvislosti položky z druhého prvku, a toto přiřazení je z jakéhokoliv hlediska nepřijatelným. Za typické odděleně sázené závislé prvky uveďme: obrázek auta a jeho SPZ, autora příspěvku a jeho univerzitu, citát a jeho autora, citát a původ citátu, úryvek knihy a název knihy nebo logo univerzity a kontaktní osobu, jméno a příjmení, rodiče a iniciály dítěte, knihu a její `ISBN`, zemi a její měnu, oficiální jazyk a vlajku země, atd.

Pokud sázíme na jednom místě záhlaví nebo zápatí jméno a příjmení dohromady, jsou sice logicky spojené, ale nejsou sázeny odděleně a tudíž je nepovažujeme za závislé prvky.

Uveďme si konkrétní příklad sazby logicky provázaných prvků sázených odděleně. Mějme dva autory, a to pana Jiřího Hlavsu s článkem *E-learningová omezení* a paní Helenu Mokrou s článkem *Tvorba univerzitní šablony.*

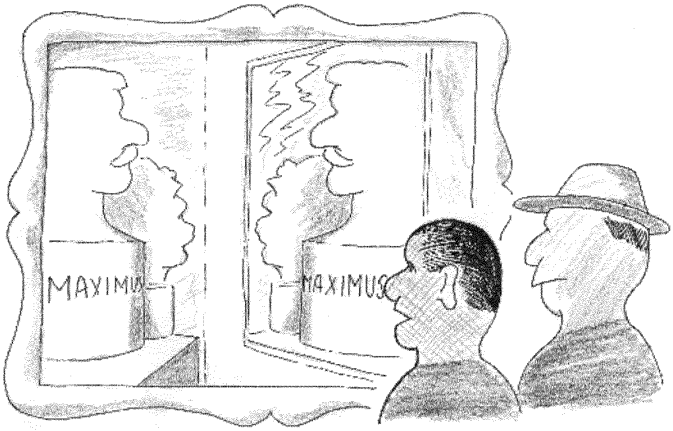
```
\fancyhead[C]{\ifcase\value{page}\or
Jiří Hlavsa\or
Helena Mokrá\else \relax \fi }
\fancyfoot[C]{\ifcase\value{page}\or
E-learningová omezení\or
Tvorba univerzitní šablony\else \relax \fi }
```

Pokud u prvního proměnného prvku (jméno autora) chci tyto dva autory prohodit, dostáváme:

```
\fancyhead[C]{\ifcase\value{page}\or
Helena Mokrá\or                                  %Zde nastala změna.
Jiří Hlavsa\else \relax \fi }                    %Zde nastala změna.
\fancyfoot[C]{\ifcase\value{page}\or
E-learningová omezení\or
Tvorba univerzitní šablony\else \relax \fi }
```

Teď je však naše záhlaví a zápatí chybné, protože paní Mokrá nikdy článek *E-learningová omezení* nenapsala. Podobně pan Hlavsa neposlal příspěvek *Tvorba univerzitní šablony.* Proto také musíme prohodit dvě položky ve druhém proměnném prvku.

Je to řešení spíše nepřijatelné, zvláště když si uvědomíme, že těchto stránkově závislých proměnných prvků můžeme mít na straně poměrně hodně, nepočítaje množství nastalých změn.



Malíř uměl rozhodně zrcadlově kreslit,
ale neuměl zrcadlově psát!

Grafická vsuvka: *Jak na zrcadlovou sazbu?*

6. Generování pomocí PHP a MySQL

Nechme raději soubor `zahlavi.tex` vhodně vygenerovat. Uveďme si ukázku pomocí programu v `PHP` zpracovávající data z `MySQL`, ve které máme uloženy potřebné informace (včetně uložených odkazů na galerie obrázků, rodinných fotek, apod.).

Při přidání, mazání a jiném řazení položek necháme soubor znovu vygenerovat a TEX má neustále aktuální vstupní soubor `zahlavi.tex`.

6.1. Program `meetings.php`

Připravíme si soubor `meetings.php` a ten přes webový prohlížeč spustíme. Soubor vypadá např. takto:

```
190 <?php
191 @$spojeni=mysql_connect("localhost","spravce","heslo_spravce");
192 @mysql_select_db("conference");
193 @$result=mysql_query("SELECT * FROM papers");
194 $mp=fopen("zahlavi.tex","w");
195 fwrite($mp,"%Dne: ".date("F d Y H:i:s.",getlastmod())."\r\n");
196 fwrite($mp,"\\fancyhead[R]{\\ifcase\\value{page}\\or\r\n");
197 while($line = mysql_fetch_array($result)) {
198    fwrite($mp,$line["article"]);
199    fwrite($mp,"\or \r\n"); }
200 fwrite($mp,"\\relax\\else\\relax\\fi }\r\n");
201 mysql_data_seek($result, 0);
202 fwrite($mp,"\\fancyfoot[R]{\\ifcase\\value{page}\\or\r\n");
203 while($line = mysql_fetch_array($result)) {
204    fwrite($mp,$line["authors"]);
205    fwrite($mp,"\or \r\n"); }
206 fwrite($mp,"\\relax\\else\\relax\\fi }\r\n");
207 pclose($mp);
208 @mysql_close($spojeni);
209 echo "Soubor zahlavi.tex byl vygenerován: ";
210 echo date ("F d Y H:i:s.", getlastmod()); ?>
```

Program `meetings.php` se řádkem 191 připojí k nyní fiktivnímu serveru, řádkem 192 k databázi `conference`, a žádá řádkem 193 o informace z tabulky `papers`. Řádkem 195 do souboru `zahlavi.tex` vypíše informace o vzniku tohoto souboru, na řádcích 196 – 200 generuje záhlaví a na řádcích 202 – 206 pak generuje zápatí. Další řádky programu jsou uzavírající sekvence a kontrolní výpis pro uživatele o dokončení běhu programu.

Jediným problémem snad může být kódování, ale ledacos se dá napravit při využití příkazu `mb_convert_encoding`. Není však problém využít externího převaděče a uzpůsobit kódování výstupu z `MySQL` pro potřeby TEXu před vlastním načtením tohoto souboru.

6.2. Výstup z programu `meetings.php`

Výstupem je soubor `zahlavi.tex`, který může být poměrně velký. To platí i pro náš příklad, a proto uvádíme vždy první a poslední záznam a řádky 214 a 219, a nahrazujeme vnitřní údaje třemi tečkami. Ukázka z databáze o konferenci může vypadat následujícím způsobem:

```
211 %Dne: February 02 2005 05:41:52.
212 \fancyhead[R]{\ifcase\value{page}\or
213 Internet Interfaces\or
214 %...                Zde následuje druhý až předposlední záznam.
215 Exact Procedures : Two Independent Samples\or
216 \relax\else\relax\fi }
217 \fancyfoot[R]{\ifcase\value{page}\or
218 Gustav Horník\or
219 %...                Zde následuje druhý až předposlední záznam.
220 Saskie Burešová\or
221 \relax\else\relax\fi }
```

Na stránkách, počínaje první, se v záhlaví vpravo objeví název příspěvku a v zápatí vpravo jméno autora. Na první straně bude uveden příspěvek *Internet Interfaces* od pana Horníka a jako poslední záznam, pokud strana existuje, bude uveden příspěvek *Exact Procedures : Two Independent Samples* od paní Burešové. Pokud je stran dokumentu více než je položek v databázi, tak se provádí `\relax`.

Pokud chceme udělat změny, uděláme je přímo v databázi a necháme znovu vygenerovat `zahlavi.tex` spuštěním `meetings.php`. Tím se nejen berou aktuální údaje, ale uloží se do poznámky i doba, kdy se soubor `zahlavi.tex` vygeneroval.

Přímo v `MySQL` lze připravit generující skript za pomocí příkazu `concat`. V tabulkových procesorech je na přípravu souboru `zalozka.tex` jednou z alternativ příkaz `concatenate`. Slouží ke spojování různých řetězců v jeden textový, což se může hodit zvláště k napojení příkazu `\or`.

Pokud máme v databázi i texty určené do těla dokumentu, můžeme je pomocí `PHP` také vygenerovat. Velkým pomocníkem je pak příkaz TEXu `\newpage`.

6.3. Program `rozmisti.php`

Pomocí nástrojů `PHP` a `MySQL` můžeme udělat řadu jiných vzorových řešení. Ukážeme si řešení následující situace. Máme přepsány texty v databázi a chceme je rozmístit tak, aby první položka byla na první straně v záhlaví, druhá v jejím zápatí, třetí na druhé straně v záhlaví, čtvrtá pak v zápatí druhé strany, atd. Podobně uvažujeme o rozmístění do rohů stránek, na okrajích dokumentu ve směru hodinových ručiček, atd.

Rozšiřme a připravme si program `rozmisti.php` následující cestou:

```
222 <?php @$spojeni=mysql_connect("PC","uzivatel","heslo");
223 @mysql_select_db("umeni");
224 @$result=mysql_query("SELECT * FROM basen001");
225 $mp=fopen("zahlavi.tex","w");
226 fwrite ($mp,"%".date ("F d Y H:i:s.", getlastmod())."\r\n");
227 fwrite ($mp,"\\setlength{\\headheight}{24pt}\r\n");
228 fwrite($mp,"\\renewcommand{\\headrulewidth}{2.4pt}\r\n");
229 fwrite($mp,"\\renewcommand{\\footrulewidth}{1.6pt}\r\n");
230 fwrite($mp,"\\fancyhf{}\r\n");
231 fwrite($mp,"\\lhead{\\ifcase\\thepage\\or\r\n");
232 $citac=0;while ($line = mysql_fetch_array($result)) {
233 if($citac==0){
234    fwrite($mp,"\\emph{".$line["radkove"]."}");
235    fwrite($mp,"\or \r\n"); $citac++;}
236       else {$citac=0;} }
237 fwrite($mp,"\\relax\\else\\relax\\fi }\r\n");
238 mysql_data_seek($result, 0);
239 fwrite($mp,"\\lfoot{\\ifcase\\thepage\\or\r\n");
240 $citac=1;
241 while ($line = mysql_fetch_array($result)) {
242 if($citac==0){
243    fwrite($mp,"\\emph{".$line["radkove"]."}");
244    fwrite($mp,"\or \r\n"); $citac++; }
245       else {$citac=0;} }
246 fwrite($mp,"\\relax\\else\\relax\\fi }\r\n");
247 pclose($mp); @mysql_close($spojeni);
248 echo "Verze ze dne: ".date ("F d Y H:i:s.", getlastmod()); ?>
```

Jistě bychom našli řadu programátorských postupů a fint, jak se s takovým generováním TEXových zdrojů zabývat i mnohem hlouběji. V tomto programu jsme na řádku 227 navíc nastavili i pracovní velikost záhlaví, protože úryvky básní jsou dvouřádkové.

6.4. Výstup z programu `rozmisti.php`

Výstupem jsou zdravotní napomenutí z kapitoly *Lokšan kontra Griespek*[2] a soubor `zahlavi.tex` vypadá takto:

```
249 %February 06 2005 02:34:10.
250 \setlength{\headheight}{24pt}
251 \renewcommand{\headrulewidth}{2.4pt}
252 \renewcommand{\footrulewidth}{1.6pt}
```

[2] Rudolf Richard Hofmeister. *Březnické obrázky.* Šolc a Šimáček, Praha 1932.

```
\fancyhf{}
\lhead{\ifcase\thepage\or
\emph{Ledna pouštění nech, ráno\\\ \ \ %
   traňk pí, potom dobré víno.\ ---}\or
\emph{Března střídmý jdi do lázně,\\\ \ \ %
   pulej pí, a drž se vážně.\ ---}\or %
%...                                  Měsíce by zde pokračovaly.
\relax\else\relax\fi }
\lfoot{\ifcase\thepage\or
\emph{Února pusť z palce, mej se,\\\ \ \ %
   hoduj, zvěř jez, hry varuj se.\ ---}\or
\emph{Dubna přijmi počištění,\\\ \ \ %
   traňk pí, pouštěj, jez koření.\ ---}\or
%...                                  Měsíce by zde pokračovaly.
\relax\else\relax\fi }
```

Vysázené záhlaví a zápatí by vypadalo následujícím způsobem:

Strana 1	záhlaví	*Ledna pouštění nech, ráno* *traňk pí, potom dobré víno.* —
		TĚLO DOKUMENTU
	zápatí	*Února pusť z palce, mej se,* *hoduj, zvěř jez, hry varuj se.* —
Strana 2	záhlaví	*Března střídmý jdi do lázně,* *pulej pí, a drž se vážně.* —
		TĚLO DOKUMENTU
	zápatí	*Dubna přijmi počištění,* *traňk pí, pouštěj, jez koření.* —

⋮

7. Generování záložky na okraji

Tím, že jsme použili PHP, nemůžeme říci, že TEXem nelze nic generovat. Jak se budeme moci přesvědčit, tak to lze – samotný TEX však pro mě spíše není databázovým systémem na webové záležitosti. Můžete však se mnou nesouhlasit.

Budeme generovat okrajovou záložku tímto způsobem. Nejprve si potřebné údaje při prvních bězích TEXu vygenerujeme do souboru. Při závěrečných bězích TEXu si tento soubor načítáme a údaje z něj necháme vysázet. Okrajovou záložku budou tvořit části a sekce, a aktuální jedna nebo dvě sekce budou na dvoustraně

orámovány. Sazba se provede vpravo shora dolů. Velmi podobně lze připravit sazbu vlevo na dvoustraně, na každé straně, atd.

7.1. Definování proměnných

Začněme jedinou nezautomatizovanou proměnnou `posun`. Pomocí ní TEXu říkáme, zda-li jsme v úvodních (někdy jen v prvním) nebo již v závěrečných bězích TEXu (obecně můžeme říci: v posledním běhu TEXu).

```
\newcount\posun
\posun=1 %Implicitně je jedna. Pro vysázení měníme na \posun=0.
```

Pokud nejsme v závěrečném běhu TEXu, nechávejme `\posun=1`. Ve chvíli kdy končíme práci na dokumentu, dejme místo jedničky jinou hodnotu, např. `\posun=0`.

Na této proměnné závisí řada dějů. Jedním z nich je otevření a možný zápis do souboru `zalozka.tex`:

```
\ifnum\posun=1
\immediate\newwrite\tak%
\immediate\openout\tak zalozka.tex \fi
```

Definujeme proměnnou `\podel` na zjištění sudé/liché strany. Dále si založíme proměnné `stranapr`, `zaznampr` a `aktualni`, pomocí nichž řídíme vlastní proces generování a zápisu do souboru.

```
\newcount\podel \newcount\stranapr
\newcount\zaznampr \zaznampr=0
\newcounter{aktualni} \setcounter{aktualni}{1}
```

Jak budeme moci vypozorovat dále, první a poslední bloky zápisů jsou rozdílné vůči ostatním. To realizujeme nastavením proměnné `\zaznampr=0`.

7.2. Definování příkazů

Příkaz `\zapsano` slouží k formulaci výstupní procedury:

```
\newcommand\zapsano{\Roman{aktualni}--\thesection}
```

Dále potřebujeme rozšířit příkaz `\section`. Tento způsob není nejjednodušší a vyžaduje více zkušeností s TEXem.

Lze to obejít příkazem, který bude stát mimo příkaz `\section` a nebude na něm přímo závislý. Nazvěme ho `\gnadpis` a definujme jej tímto způsobem:

```
\newcommand\gnadpis{\newpage \ifnum\posun=1%
\stranapr=\value{page}%
\ifodd\stranapr\advance\stranapr by -2
   \else\advance\stranapr by -1\fi%
\ifnum\zaznampr=0%
```

```
282    \immediate\write\tak{\string\i fnum\string\t hepage>%
283    \the\stranapr} \zaznampr=1\podel=\value{page}%
284    \ifodd\podel\advance\podel by 1\else\advance\podel by 2\fi%
285 \else%
286    \podel=\value{page}\advance\podel by -1
287    \ifodd\podel\advance\podel by 1\else\advance\podel by 2\fi%
288    \immediate\write\tak{\string\i fnum%
289    \string\t hepage<\the\podel
290    \string\f ramebox{\zapsano} \string\e lse \zapsano \ %
291    \string\f i\string\e lse \zapsano}%
292    \immediate\write\tak{\string\f i}%
293    \immediate\write\tak{\string\i fnum%
294    \string\t hepage>\the\stranapr}
295    \setcounter{aktualni}{\value{part}}
296 \fi \fi }
```

Na řádku 281 rozlišujeme, zda-li se jedná o náš první zápis nebo o další. První zápis formulují řádky 282 a 283, další se řídí řádky 288 až 294. Zápis zpětného lomítka není ideálním řešením, ale splňuje to svůj účel.

Příkaz `\gnadpis` tedy zajišťuje výstup do souboru `zalozka.tex`, a to vyjma závěrečných sekvencí.

Poslední zápis pro testování přítomnosti nadpisu na dvoustraně definujeme zvlášť jako příkaz `\poslednipr`:

```
297 \newcommand\poslednipr{\ifnum\posun=1%
298 \podel=\thepage \advance\podel by 1%
299 \immediate\write\tak{\string\i fnum\string\t hepage<\the\podel
300 \string\f ramebox{\zapsano}\string\e lse \zapsano
301 \ \string\f i\string\e lse \zapsano}%
302 \immediate\write\tak{\string\f i}%
303 \immediate\closeout\tak \fi}
```

Tento příkaz použijeme jen jednou, a to na poslední straně, u které ještě zkoumáme přítomnost nadpisu, a kde povolujeme možnost vysázení záložky. Pokud tento příkaz nepoužijeme, bere se za poslední testovanou stranu poslední strana dokumentu. Pomocí řádku 303 uzavíráme soubor `zalozka.tex`.

7.3. Vygenerovaný soubor `zalozka.tex`

Uveďme si jednu ukázku po vlastním generování. Na straně 9 máme sekci první, druhou máme na straně 11 a nadpis třetí sekce je na straně 19. Balíkem `fancyhdr` načítaný soubor může již po prvním běhu vypadat např. takto:

```
304 \ifnum\thepage>7
305 \ifnum\thepage<12\framebox{I--1} \else I--1\ \fi\else I--1
```

```
\fi
\ifnum\thepage>9
\ifnum\thepage<20\framebox{I--2} \else I--2\ \fi\else I--2
\fi
\ifnum\thepage>17
\ifnum\thepage<22\framebox{II--3}\else II--3\ \fi\else II--3
\fi
```

U každého `\zapsano` byla zjišťována přítomnost na dvoustraně a zapsána přes příkazy `\ifnum`. Pokud je sázena záložka a zjištěna přítomnost na dvoustraně, `\zapsano` je zarámováno vůči těm, které na dvoustraně nejsou. Poněvadž každá sekce je vysázena na nové straně, musí na straně předchozí být také předchozí sekce. U částí se používá vlastní styl a nemusí nás tedy trápit.

7.4. Styl `zalozka`

Pro snazší přepínání mezi styly si definujme styl `zalozka` takto:

```
\def\zalozka{\fancyhead[RE]{\thepage}%
\fancyhead[RO]{\thepage\ifnum\posun=0%
\begin{picture}(0,0)\put(3,-688){%
\rotatebox[origin=r]{270}{\input{zalozka.tex}}
} \end{picture} \fi } }
```

7.5. Použití v hlavním dokumentu

Přípravu pomocného souboru řídíme příkazem `\posun=1` nebo `\posun=0`. Pokud má být záložka správně vysázena, musí dokument proběhnout TEXem minimálně dvakrát. Poprvé a minimálně jednou pro `\posun=1` a podruhé (v závěrečném běhu TEXu) a minimálně také jednou pro `\posun=0`.

Dokument může vypadat následujícím způsobem:

```
\documentclass[twoside]{article}
\usepackage{graphicx} \usepackage{fancyhdr} \pagestyle{fancy}
%...                     Definování všech příkazů z této sekce.
%...   Nastavení příkazu \posun=1 a na posledním běhu \posun=0.
\zalozka                     %Spuštění námi připraveného stylu.
\begin{document}
\part{První...}\thispagestyle{empty}
\gnadpis\section[Ada]{První...}           %V ukázce na str.  9.
\gnadpis\section[Eda]{Druhá...}           %V ukázce na str. 11.
\part{Druhá...}\thispagestyle{empty}
\gnadpis\section[Oto]{Třetí...}           %V ukázce na str. 19.
\poslednipr
```

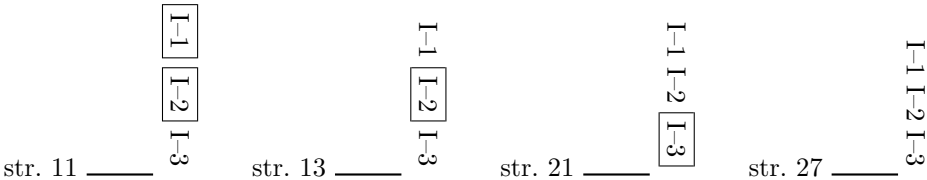
```
\section{Čtvrtá...}          %Tuto sekci nechci mít v záložce.
%...                         Následují další pasáže dokumentu.
\end{document} \bye
```

V českém dokumentu lze do preambule přidat `\usepackage{czech}` a nastavit si vstupní kódování, atd. To není náplní tohoto příspěvku. Nastavené x a y u příkazu `\put` jsou pevné a spíše méně vhodné, ale nechávám na čtenáři jejich případnou modifikaci, příp. využití zápatí k vysázení záložky. Příkaz `\gnadpis` je nastaven tak, aby byl vždy před příkazem `\section`. Jsme tedy v oblasti proměnných prvků, ale s určitým krokem zobrazování.

7.6. Záložka na dvoustraně

Tato záložka je připravena tak, že každá změna je řízena příkazem `\gnadpis` a všechny odhalené `\zapsano` se vysází. Jen aktuální sekce na dvoustraně budou zarámované pomocí `\framebox`.

Horizontální linky nám v tomto případě oddělují tělo dokumentu od jeho zápatí. Následují ukázky z několika stran při načtení zmíněného `zalozka.tex`:



Na dvoustraně se stranou 11 se vyskytují texty dvou sekcí, na straně 13 a 21 pak texty jen jedné ze sekcí.

Strana 27 je zvlášť zajímavá, protože to je strana, která následovala až po zadání `\poslednipr`. Mohou to být přílohy, doplňky, atd., které k textům s nadpisy nepatří. Vytvořená záložka je vysázena, ale žádná sekce není zarámována.

Pozorný čtenář si již sám jistě upraví záložku položek vysázených na jedné straně.

Pro hloubavého čtenáře: Čtenář může dle chuti vyzkoušet svůj um, ať už se jedná o pevné umístění položek záložky, anebo umístění proměnně závislé od počtu částí, sekcí nebo počtu stran dokumentu. První nechť se vysází nahoře u záhlaví a pak rovnoměrně dle délky těla dokumentu až po poslední, která může být dole těsně u zápatí.

Např. sekce třetí (ze čtyř) na straně 25 (z celkových 125) se vysází přesně ve třech čtvrtinách (sazba závislá na počtu kapitol; 3/4) nebo v jedné pětině (sazba závislá na stránkovém umístění; 25/125) délky mezi záhlavím a zápatím.

8. Souhrnná ukázka

Ukažme si ještě jeden příklad. Ten také využije balíčku `fancyhdr` a bude postupně obsahovat informace v záhlaví (text) a zápatí (grafiku) z `MySQL` databáze s externími obrázky za pomoci vygenerovaného souboru skriptovacím jazykem `PHP`, jednoduchou ukázku dekorace (chceme-li ornamentů) mimo oblast záhlaví a zápatí a se záložkou na stranách. V této souvislosti se dále používá sazba pasovacích, ořezových a zaměřovacích prvků.

Záhlaví a zápatí se generuje z databáze, kde máme uloženy citáty a odkazy na `.eps` soubory například s japonskými ekvivalenty. Soubory by měly být uloženy v adresáři s hlavním dokumentem, anebo při generování nastavíme absolutní cestu.

Záložka je téměř samogenerující a zobrazuje se na lichých stránkách. Dekorace už jen vhodně přidáme pomocí prostředí `picture` dle našeho grafického cítění.

V prvním kroku si vhodně upravíme generující `.php` soubor, a to touto cestou:

```
<?php
@$spojeni=mysql_connect("localhost","root","triadpass");
@mysql_select_db("skripta");
@$result=mysql_query("SELECT * FROM citaty");
$mp=fopen("zahlavi.tex","w");
fwrite($mp,"%".date ("F d Y H:i:s.", getlastmod()).".\r\n");
fwrite($mp,"\\renewcommand{\\headrulewidth}{1.2pt}\r\n");
fwrite($mp,"\\renewcommand{\\footrulewidth}{0.8pt}\r\n");
fwrite($mp,"\\fancyhf{}\r\n");
fwrite($mp,"\\lhead{\\ifcase\\thepage\\or\r\n");
while ($line = mysql_fetch_array($result)) {
   fwrite($mp,"\\emph{".$line["english"]."}");
   fwrite($mp,"\or \r\n"); }
   fwrite($mp,"\\relax\\else\\relax\\fi }\r\n");
mysql_data_seek($result, 0);
fwrite($mp,"\\lfoot{\\ifcase\\thepage\\or\r\n");
while ($line = mysql_fetch_array($result)) {
   fwrite($mp,"\\includegraphics[height=10pt]{");
   fwrite($mp,$line["japanese"].".eps}");
   fwrite($mp,"\or \r\n"); }
   fwrite($mp,"\\relax\\else\\relax\\fi }\r\n");
pclose($mp);
@mysql_close($spojeni);
echo "Last modified: ".date ("F d Y H:i:s.", getlastmod()); ?>
```

Takto připravený `.php` soubor spustíme a výstupem je soubor `zahlavi.tex`, který vypadá následujícím způsobem (opět vnitřní položky vynecháváme, tedy řádky 363 a 368):

```
357 %February 05 2005 01:14:40.
358 \renewcommand{\headrulewidth}{1.2pt}
359 \renewcommand{\footrulewidth}{0.8pt}
360 \fancyhf{}
361 \lhead{\ifcase\thepage\or
362 \emph{Empty barrels make the most noise.}\or
363 %...      Zde následují další položky načtené z MySQL databáze.
364 \emph{Security is the greatest enemy.}\or
365 \relax\else\relax\fi }
366 \lfoot{\ifcase\thepage\or
367 \includegraphics[height=10pt]{empty.eps}\or
368 %...      Zde následují další položky načtené z MySQL databáze.
369 \includegraphics[height=10pt]{security.eps}\or
370 \relax\else\relax\fi }
```

Nyní je téměř vše připraveno pro výsledný TEXový dokument. Ten strukturujeme takto:

```
371 \documentclass[twoside]{article}
372 %...                        Načteme minimálně tyto tři balíčky.
373 \usepackage{graphicx} \usepackage{fancyhdr} \pagestyle{fancy}
374 \input{zahlavi.tex}   %Soubor z předchozí strany tohoto článku.
375 %...                Načteme čtvrtý zmiňovaný styl ze sekce 2.6.
376 %...          Načteme zdrojové řádky ze sekcí 7.1., 7.2. a 7.4.
377 %  V načtených řádcích přepínáme proměnnou \posun=1 a \posun=0.
378 \zalozka      %Rozšíříme styl zavedený v zahlavi.tex o záložku.
379 \fancyhead[C]{\begin{picture}(0,0)\put(0,15){
380 \rotatebox[origin=r]{180}{\includegraphics%
381    [width=6.9cm, height=0.5cm]{obr1}} }
382 \put(-200,15){\includegraphics[width=7cm, height=0.5cm]{obr1}}
383 \end{picture} }    %Prostředí picture rozšiřuje styl o grafiku.
384 \begin{document}
385 \urovenA{Vítejte!}         %Úroveň part. Bez zápisu do záložky.
386 \gnadpis\urovenB{Sazba záhlaví} %Zápis záložky. Úroveň section.
387 \gnadpis\urovenB{Sazba záložky} %Zápis záložky. Úroveň section.
388 \gnadpis\urovenB{Volení stylů}  %Zápis záložky. Úroveň section.
389 \urovenC{O stylu 1}  %Bez zápisu do záložky. Úroveň subsection.
390 \urovenC{O stylu 2}  %Bez zápisu do záložky. Úroveň subsection.
391 \poslednipr    %Uzavření záložky. Nenásledují příkazy \gnadpis.
392 \end{document}
```

Výstup našich snah může vypadat například takto:

Co se stane, pokud máme takto načtených přísloví méně než je stran dokumentu? V dalších řádcích se provádí `\relax`, tedy v záhlaví nebo zápatí na dalších stranách se nezobrazuje nic nového. Co se stane, pokud máme načtených řádků více než je stran dokumentu? Překladač o nich ví, ale nemá příležitost je v dokumentu zobrazit. Zobrazí se nanejvýš tolik přísloví, kolik je stran dokumentu.

Zdrojový kód `ukazka.tex` a ukázky v souborech `ukazka.pdf` a `ukazka.ps` hledejte na serveru CSTUGu. Dále zde najdete vygenerovaný a uměle vytvořený soubor `zahlavi.tex` a předpřipravený soubor `zalozka.tex`. Grafický doplněk je vytvořen pomocí souboru `obr1` a poslední přiložené soubory jsou `empty.eps` a `security.eps`.

9. Slovo závěrem

Zajímavých záhlaví, zápatí a záložek se dá připravit určitě celá řada a budu rád, pokud mi na ukázku svá nebo i cizí díla, případně odkazy na ně, pošlete. To samozřejmě platí pro libovolná „živá záhlaví a zápatí“, jak tento termín někteří typografové používají, včetně vždy vítaných neřešených kousků.

Stejné formátovací prvky, řadu dalších textů a především styl mluvy se čtenářem lze očekávat v budoucnosti u připravovaných skript *DTP a elektronické publikování* a *DTP and Electronic Publishing* pro stejnojmenné předměty na Fakultě managementu a ekonomiky Univerzity Tomáše Bati ve Zlíně.

Post mortem (Ponaučení)

Doufám, že čtenář tiše prominul, ale osobně velice rád čísluji nejlépe vše a všude. Zvláště to platí u řádků zdrojových kódů, na které se odkazuji pomocí balíčku `lineno`. Pokud bych tyto řádky nečísloval, nebyl by tento *Post mortem* napsán.

Číslování přináší problém v podobě rychlého použití z `.ps` nebo `.pdf` souboru a přetažení do svého zdrojového dokumentu, protože číslo se umístí v `.txt` souboru pod příslušný řádek (extrahováno z `.ps`) nebo na konci řádku (extrahováno z `.pdf`). To vyžaduje další nutné úpravy (např. opět pomocí skriptovacího jazyka `PHP`), než si čtenář může zdrojový kód ve svém dokumentu vyzkoušet. Nemluvě o úporném přepisu z tištěného podkladu.

Poněvadž zdrojové kódy píši strukturovaně, vždy na začátku řádku zadaným příkazem `\verb` a na konci ukončené znakem | nebo příkazem `\linelabel` (příkazem `\verb` textový odstavec nikdy nezačínám), lze tyto údaje vyextrahovat z `.tex` dokumentu za pomoci `PHP`. Pomocí čítače můžeme za zdrojový řádek přidat informaci o čísle řádku v příspěvku, ale již jen jako poznámku.

Bloky zdrojových kódů začínají `\linenumbers` a jsou ukončeny příkazem `\citk`, který definujeme takto:

```
393 \def\citk{\nolinenumbers\medskip}
```

Toho můžeme využít a ve výstupním souboru přidat pro přehlednost volný řádek. Program jsem nazval `extrakce.php` a vypadá takto:

```
394 <?php
395 $vstup=fopen("zdroj.tex","r"); $zapis=fopen("kod.tex","w");
396 $citac=0; $delka=0;
397 while (!feof($vstup)) {$radek=fgets($vstup); $label="";
398 if (substr($radek,0,6)=="\verb".chr(124)){
399    $label=strrchr($radek,chr(124)); }
400 if (substr($radek,0,5)=="\citk"){
401    echo "<br>"; fwrite($zapis,"\r\n"); }
402 if ($label!=""){$citac++;
403    $delka=strlen($radek)-strlen($label)-6;
404    echo substr($radek,6,$delka)."%".$citac."<br>";
405    fwrite($zapis,substr($radek,6,$delka)."%".$citac."\r\n"); }
406 } fclose($vstup); fclose($zapis); ?>
```

Soubor `zdroj.tex` je zdrojový kód tohoto článku nebo jakýkoli dokument se stejnou strukturou u číslovaných řádků zdrojových kódů. Soubor `kod.tex` je vygenerovaným souborem a konkrétně pro tento článek je zpřístupněn na serveru CSTUGu. Pokud změníme na řádku 395 výstupní soubor na `kod_bez.tex` a z řádků 404 a 405 odstraníme `"%".$citac.`, dostaneme soubor bez komentářů. To je výhodné u `PHP`, protože zde se nekomentuje pomocí znaku procenta.

Nechť samotný článek a další zveřejněné soubory dobře slouží.

Slovníček pojmů

Uveďme si vysvětlení některých pojmů použitých v textu.

Dekorace. Výzdoba zkrášlující sazbu textu.
Ornament. Grafický prvek zkrášlující sazbu textu.
Ořezové značky. Pomůcky pro správné nařezání papíru. Typickým příkladem je příprava vizitek a A5 bulletinů (z vytištěných papírů A4).
Pasovací značky. Zobrazují postupné zvětšování mezery mezi stránkami pro vnější archy svazečku tak, aby po ořezání stránky lícovaly. Citováno z [2].
Položka proměnného prvku. Nejčastěji jedna položka z databáze, která se sází v textu právě jednou, a to nejčastěji za pomoci příkazu `\ifcase`.
Proměnný prvek. Je to takový typografický objekt, který se se změnou čítače `page`, obecně však vlivem jakéhokoliv čítače, mění v sázeném textu.
Prvek. Objekt. Typografický prvek. Proměnný prvek. V podstatě jakákoli diskutovaná pasáž sazby.
Stránkově nezávislé. Jakákoli položka z proměnného prvku může být umístěna v záhlaví nebo zápatí na libovolné straně a neztratí se přitom významové souvislosti mezi položkami proměnných prvků.
Stránkově závislé. Nelze vysázet položku jednoho proměnného prvku na jinou stranu, aniž by se položky jiných proměnných prvků v záhlaví nebo zápatí bez ztráty významu musely také přesouvat.
Zaměřovací značky. Slouží ke správné identifikaci (umístění a natočení) prvků na straně. Typickým příkladem je skenování testů pro jejich automatické vyhodnocení.
Živé záhlaví a zápatí. Jakákoli sazba v záhlaví nebo zápatí, která má čtenáři usnadnit orientaci v publikaci.

Slovníček Top 40

Uveďme si některé pojmy ve stylu EN→CZ.

advance [ədˈvɑːns] . . . . . rozšířit
alpha [ælfə] . . . . . abecedně
arabic [ˈærəbɪk] . . . . . arabsky
article [ˈɑːtɪkəl] . . . . . článek
box [bɒks] . . . . . schránka
conference [ˈkɒnfərəns] . . . . . konference
connect [kəˈnekt] . . . . . připojit se
count [kaʊnt] . . . . . spočítat
decoration [dekəˈreɪʃən] . . . . . výzdoba
dependent [dɪˈpendənt] . . . . . závislý
divide [dɪˈvaɪd] . . . . . podělit
empty [ˈemptɪ] . . . . . prázdný
even [ˈɪːvən] . . . . . sudý
fancy [ˈfænsɪ] . . . . . pestrý
footing [ˈfʊtɪŋ] . . . . . zápatí
heading [ˈhedɪŋ] . . . . . záhlaví
height [haɪt] . . . . . výška
immediate [ɪˈmɪːdɪət] . . . . . okamžitě

independent [ɪndɪˈpendənt] .. nezávislý
leftmark [leftmɑːk] ..... levá značka
movie [ˈmʊːvɪ] ..... sekvence obrázků
multiply [ˈmʌltɪplaɪ] ..... vynásobit
odd [ɒd] ..... lichý
origin [ˈɒrɪʤɪn] ..... původní
page [peɪʤ] ..... strana
paper [ˈpeɪpəˈ] ..... příspěvek
picture [ˈpɪktʃəˈ] ..... obrázek
query [ˈkwɪərɪ] ..... dotaz
relax [rɪˈlæks] ..... odpočívat
result [rɪˈzʌlt] ..... výsledek
rightmark [raɪtmɑːk] .... pravá značka
rotate [rəʊˈteɪt] ..... pootočit
section [ˈsekʃən] ..... sekce
select [sɪˈlekt] ..... vybrat
string [strɪŋ] ..... textový řetězec
subsection [sʌbˈsekʃən] ..... podsekce
typesetting [ˈtaɪpsetɪŋ] ..... sazba
value [ˈvæljʊː] ..... hodnota
variable [ˈveərɪəbəl] ..... proměnná
width [wɪdθ] ..... šířka

Reference

[1] Tobias Oetiker, Hubert Partl, Irene Hyna and Elisabeth Schlegl. *The Not So Short Introduction to LaTeX 2ε*.
`http://www.ctan.org/tex-archive/info/lshort/`.

[2] Petr Olšák. *Jak používám TeX?* 4. ukázka: Inzertní příloha časopisu Dotek.
`http://www.olsak.net/texpraxe.html`.

[3] Petr Olšák. *TeXbook naruby.* Public edition 1996, Konvoj 1997. 468 pp, ISBN 80-85615-64-9.
`http://math.feld.cvut.cz/olsak/tbn.html`.

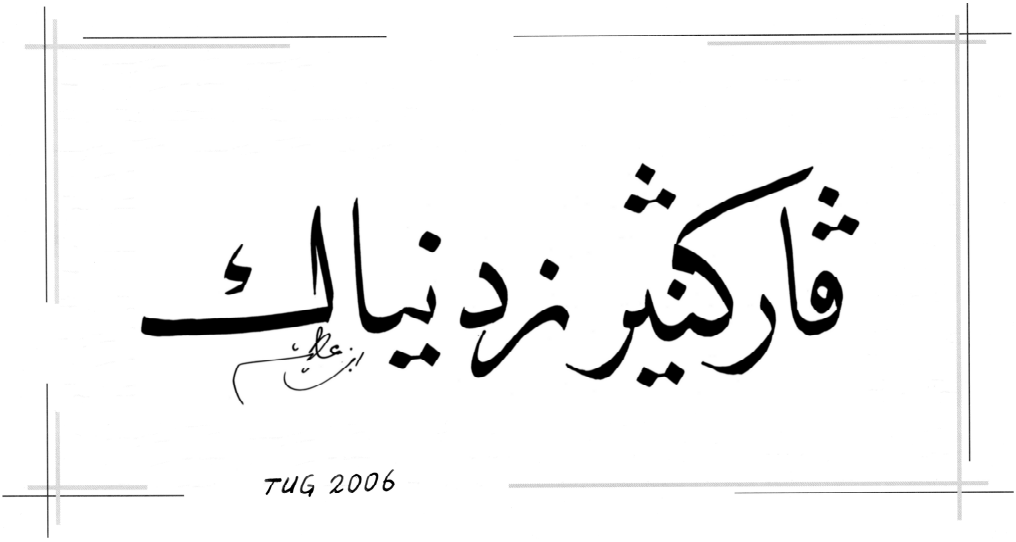
[4] Piet van Oostrum. *Page layout in LaTeX.* MAPS, 16:108 – 125, 1996.
`http://www.ntg.nl/maps/pdf/16_29.pdf`.

[5] Jozef Říha, Pavel Stříž. *TeX-Collection 2003 a 2004.* Informační Bulletin České statistické společnosti, 4(15), pp 4 – 7, Praha 2004. ISSN 1210 – 8022.
`http://www.statspol.cz/bulletiny/ib-04-4.pdf`.

Pavel Stříž, ÚIS FaME UTB ve Zlíně
`striz@fame.utb.cz`, `pavel.striz@email.cz`

27th TUG Annual Meeting, Marrákéš 2006, krátká reportáž

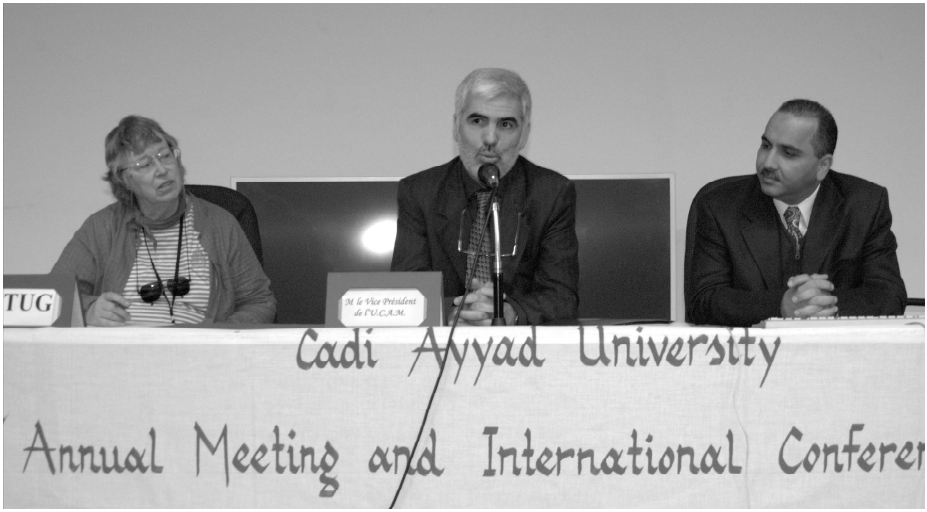
Zdeněk Wagner



Kaligrafii mého jména vytvořil Mohamed Jamal Eddine Benatia

Ve dnech 9.–11. listopadu 2006 se konalo 27. výroční shromáždění TUGu spojené s konferencí, jejíž název byl *International Conference on Digital Typography & Electronic Publishing: Localization & Internationalization*. Inspirován tématem jsem přihlásil příspěvek o hindštině v Babylónu. Překlad mého příspěvku do češtiny najdete v tomto čísle Zpravodaje. Příspěvky většiny ostatních účastníků se však týkaly arabštiny nebo arabského písma, včetně přednášky dalšího českého zástupce, Karla Píšky. Někteří účastníci žertovně říkali, že jsem jediný hindsky mluvící člověk na arabské konferenci.

Na konferenci zazněly též přednášky, které s arabštinou nesouvisely. Jonathan Kew představil XeTeX, Taco Hoekwater a Hans Hagen promluvili ve svých přednáškách o vývoji LuaTeXu, ConTeXtu, METAPOSTu a o OpenMath. Dalšími tématy byly např. LaTeX3 (Morten Høgholm) a projekt TeX Gyre (Hans Hagen, Jerzy Ludwichowski, Volker RW Schaa). Mezi sponzory projektu TeX Gyre byl uveden i CSTUG.



Obrázek 1: Zahájení konference, zleva Barbara Beeton (American Mathematical Society), Boumedine Tanouti (UCAM), Mohssine Belkoura (FSSM)

Zde bych si dovolil malou odbočku. K zamyšlení mě vede jednak zmínka Petra Olšáka o tom, že František Štorm přirovnal CSTUG ke „sdružení uživatelů kladívka“, a také *Anketa uživatelů TEXu* zveřejněná ve Zpravodaji 3–4/2003 (Josef Polách). Podle výsledků ankety poměrně mnoho Čechů a Slováků používá TEX, ale nemají potřebu se organizovat. Mezi komerčním softwarem a TEXem je však zásadní rozdíl. Komerční program si koupíte a jeho cena slouží mimo jiné jako mzda programátorům. O tom, jaké funkce bude program mít, rozhodují peníze zákazníků. Programátor implementuje to, co hodlá dostatečně velké množství zákazníků zaplatit. TEX a související software je většinou dílem dobrovolníků, kteří pracují zdarma ve volném čase. Tímto způsobem však nelze vytvořit vše. Na větší projekty musí programátor věnovat více než jen svůj volný čas, a musí tedy být za svoji práci zaplacen. K tomu je využíváno sponzorství TEXových organizací.

Uživatelé občas namítají, že TEX bude žít i bez CSTUGu. To je samozřejmě pravda. Finanční podpora dalšího vývoje se najde i v jiných končinách světa. Něco však bez CSTUGu nebude: podpora kvalitní české a slovenské typografie v mezinárodních distribucích. Nejde jen o peníze, jde i o to, že potřebujeme, aby se do jisté míry zbytek světa podřídil malé TEXové komunitě, která nad vybraná písmena píše háčky a čárky. Zde těžko pomůže hlas jednotlivců, ale váhu má organizace, která je vidět tím, že vykonává nějakou činnost, že její členové přispívají k vývoji TEXu, že jezdí na mezinárodní TEXové konference. Kdyby

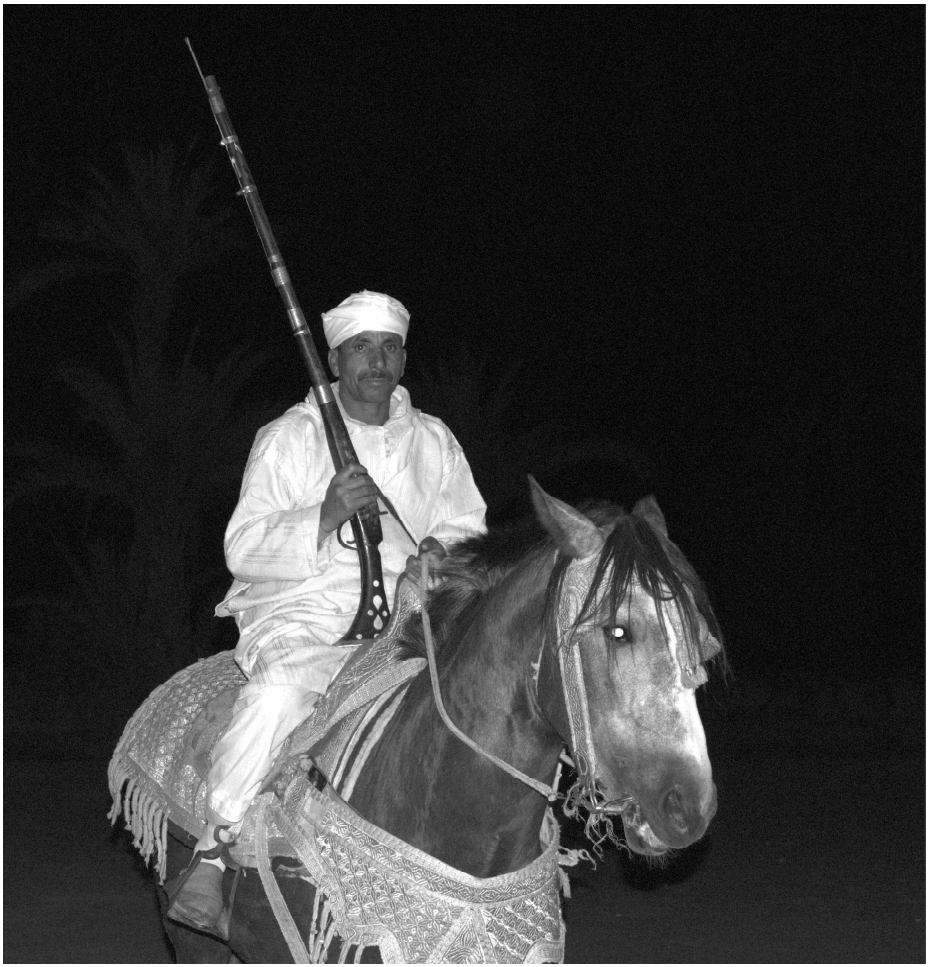
Obrázek 2: Pohled na jednu z univerzitních budov

nový český a slovenský modul pro babel nebyl oficiálním projektem CSTUGu, byla by jeho šance na zařazení do standardní distribuce LaTeXu minimální. Řada balíčků, užitečných pro Čechy a Slováky, by pak byla roztroušena po osobních webových stránkách několika aktivních TeXistů a bylo by na každém uživateli, aby si s tím poradil, jak umí.

Zanechme však chmurných vizí a vraťme se k původnímu tématu. Konference se konala na Cadi Ayyad University. Universitu není snadné fotografovat, protože na všech prostranstvích roste tropická vegetace včetně palem. Objektiv fotoaparátu zachytí tedy jen část budovy.

Konference neměla jen část TeXovou, ale neméně zajímavou část společenskou. Marrákéš je turisticky významné město. Přestože je Maroko muslimským státem, není islám praktikován nijak fanaticky. Zhruba polovina žen se obléká evropsky a mnoho z nich na ulici zdraví turisty. Přestože ortodoxní islám zakazuje zobrazování lidí i zvířat, není nutno se obávat fotografování na ulici, jen občas je nutno se smířit s tím, že někdo si za to nechá zaplatit.

Maroko bývala francouzská kolonie. Z toho vyplývá, že obyvatelé mimo svoji mateřštinu ovládají především francouzštinu. Byla ode mne dost velká troufalost, že jsem se vydal do Maroka s naprostou neznalostí arabštiny a francouzštiny.



Obrázek 3: Příjezd na banket v Chez Ami

Naštěstí jména ulic jsou psána i francouzsky, takže se průměrně gramotný turista může orientovat podle mapy. Maročané tedy většinou anglicky neumí, ale snaží se. A když si taxikář není jist, že správně rozuměl, vytáhne papír a tužku a pomůže obrázek.

Po dobu konání konference byly zajištěny obědy v malé restauraci, kde jsme dostávali spoustu dobrého jídla, že jsme to ani nestačili sníst. A závěrečnou tečkou byl grandiózní banket v Chez Ami s ohňostrojem s logem konference (viz fotografie na obálce).

Obrázek 4: Jedna z hudebních a tanečních skupin v Chez Ami

Vlastní chybou při rezervaci letenek jsem se nemohl zúčastnit výletu organizovaného následujícího dne po skončení TEXového programu. Poslední dopoledne před odletem jsem věnoval krátké procházce, kdy jsem hodlal nakoupit nějaké drobné suvenýry za zbývající dirhamy. Mezi obchůdky jsem si vybral jeden, který na mě působil nejsympatičtěji. A při placení jsem zjistil, že prodavač v tom obchodě je Ind.

Reference

TUGboat 27:2, 2006.
http://www.tug.org/TUGboat/Contents/contents27-2.html

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

Vydalo:	Československé sdružení uživatelů TEXu vlastním nákladem jako interní publikaci
Počet výtisků:	600
Uzávěrka:	2. 5. 2007
Odpovědný redaktor:	Zdeněk Wagner
Redakční rada:	Petr Aubrecht, Jiří Demel, Jaromír Kuben, Petr Sojka, Libor Škarvada, Vít Zýka
Tisk a distribuce:	KONVOJ, spol. s r. o., Berkova 22, 612 00 Brno, tel. +420 549 240 233
Adresa:	CSTUG, c/o FEL ČVUT, Technická 2, 166 27 Praha 6
Tel:	+420 224 353 611
Fax:	+420 233 332 938
Email:	`cstug@cstug.cz`

Zřízené poštovní aliasy sdružení CSTUG:

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

`board@cstug.cz`
korespondence členům výboru

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

`gacstug@cstug.cz`
grantová agentura CSTUGu

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

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

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

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

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

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