Zpravodaj Československého sdružení uživatelů TeXu

Petr Březina

Zrcadlová sazba

Zpravodaj Československého sdružení uživatelů TeXu, Vol. 18 (2008), No. 4, 212–226

Persistent URL: http://dml.cz/dmlcz/150065

Zrcadlová sazba

Petr Březina

Sázecího systému TEX lze s velkým prospěchem využívat také v klasické filologii. I zde totiž existují speciální požadavky na sazbu, o jejichž realizaci si uživatelé nejrozšířenějších textových procesorů typu WYSIWYG mohou nechat jenom zdát. Jedním z takových požadavků je tzv. zrcadlová sazba. Klasičtí filologové často chtějí, aby jejich překlady řeckých a latinských textů byly vydány v jedné knize paralelně s textem originálu: Na sudých (levých) stranách bývá originál, na lichých (pravých) překlad; přitom na každé dvojstraně musí být stejný úsek originálu a překladu. Někdy se na každou lichou stranu přímo pod text překladu připojují překladatelské poznámky, které se vztahují k místům překladu na téže straně. Na sudých stranách může být pod textem originálu kritický aparát, obsahující textové varianty.

V tomto článku bych chtěl ukázat jedno z možných řešení zrcadlové sazby.[1] Nepůjde nám o to, připravit konkrétní vzhled sazby konkrétního překladu a jeho originálu, nýbrž vytvořit skupinu maker, jejichž vhodným užitím bude moci šikovný TEXista pohodlně pořizovat zrcadlovou sazbu i značně komplikovaných dokumentů; vzhled sazby bude přitom plně v jeho rukou. Naše makra budou určena pro plain a příbuzné formáty. Důraz budeme klást na to, aby celá sazba probíhala automaticky a aby bylo možno užívat insertů obvyklým způsobem.

Základní myšlenka je tato: Každý z obou textů[2] bude uložen v jiném souboru; levý text načteme do jednoho `\vbox`u, pravý text do druhého `\vbox`u a potom budeme z těchto `\vbox`ů pomocí operace `\vsplit` postupně odkrajovat jednotlivé strany.

Kdybychom se však pokusili načíst oba texty naráz, zřejmě by došlo k překročení paměťových možností TEXu. Budeme je tedy načítat po částech tak, aby v obou `\vbox`ech (nazývejme je `\levybox` a `\pravybox`) bylo stále dost materiálu na odkrojení jedné strany. Doplňování `\vbox`ů bude probíhat v podstatě takto: Ze zdrojového souboru budeme číst jednotlivé řádky pomocí příkazu `\read`; každý přečtený řádek zkopírujeme do pomocného souboru. Jakmile narazíme na prázdný řádek (tedy zhruba řečeno na konec odstavce), přerušíme čtení a kopírování řádků, pomocí příkazu `\input` načteme pomocný soubor do příslušného `\vbox`u a poté přeměříme výšku tohoto `\vbox`u. Uvedený cyklus budeme opakovat tak dlouho, dokud výška `\vbox`u nebude mít hodnotu alespoň jeden a čtvrt `\vsize` nebo dokud nedosáhneme konce zdrojového souboru.

[1] Poznamenejme, že jiné řešení stejného úkolu bylo popsáno ve Zpravodaji č. 4 z roku 1997.

[2] To jest originál a překlad. Dále budeme oba texty označovat obecněji podle jejich polohy na dvojstraně jako „levý text“ a „pravý text“.

Občas se stává, že nějaká skupina přesahuje z jednoho odstavce do druhého; dochází k tomu např. při sazbě uvozovek pomocí makra \uv. V takovém případě je nežádoucí, aby načítání textu bylo přerušeno mezi těmito dvěma odstavci. Uživatel by samozřejmě mohl ošetřit takové případy tím, že by uvnitř skupin používal místo prázdného řádku řídicí sekvenci \par. Aby se uživatel nemusel tímto problémem vůbec zabývat, vytvoříme mechanismus, který se sám postará o to, aby načítání textu nebylo přerušeno uvnitř skupiny. Přitom budeme mít na zřeteli, že skupiny, uvnitř kterých se mohou vyskytnout prázdné řádky, nemusejí být ohraničeny jen explicitními kudrnatými závorkami, nýbrž že mohou být vymezeny také pomocí implicitních závorek alias \bgroup a \egroup, dále pomocí alternativních příkazů \begingroup a \endgroup, případně pomocí nějakých uživatelem definovaných dvojic maker.

Teď, když už víme, co by měla umět makra, která budou zajišťovat doplňování boxů \levybox a \pravybox, pustíme se do jejich definování.

```
\newread\levysoubor % zdrojový soubor s levým textem
\newbox\levybox % box, kam budeme načítat levý text
\newdimen\prevdepthL % \prevdepth na konci boxu \levybox
\let\nastavlevy=\relax % specifická nastavení pro levý text
\def\doplnlevybox{\let\next=\relax
   \ifdim\ht\levybox<1.25\vsize
      \ifeof\levysoubor \else \nactilevytext
         \let\next=\doplnlevybox \fi\fi \next}
\def\nactilevytext{\let\soubor=\levysoubor \kopiruj
   \setbox\levybox=\vbox{\break \unvbox\levybox
      \let\versnum=\versnumL \prevdepth=\prevdepthL
      \nastavlevy \input\jobname.tmp
      \global\prevdepthL=\prevdepth}%
   \setbox0=\vsplit\levybox to 0pt }
```

V rámci cyklu \doplnlevybox pracuje makro \nactilevytext. Z něj je nejdříve zavoláno makro \kopiruj, které zkopíruje ze zdrojového souboru do pomocného souboru úsek textu ohraničený prázdným řádkem; ke kódu tohoto makra se dostaneme za chvilku. Pomocný soubor je potom načten do boxu \levybox. Nastavení \prevdepth na hodnotu, která byla při minulém načítání zaznamenána do registru \prevdepthL, zajistí správnou meziřádkovou mezeru mezi odstavci. Řídicí sekvence \nastavlevy, která se aktivuje vždy před načtením pomocného souboru, je určena pro uživatele; uživatel ji může nadefinovat jako makro, které provede specifická nastavení pro levý text (nastavení správné tabulky dělení slov apod.). Řídicí sekvence \versnum a \versnumL souvisejí se synchronizací textů; o jejich přesném významu si povíme později. Nakonec jsou pomocí operace \vsplit vymazány ze začátku boxu \levybox odstranitelné elementy a místo nich je vložena mezera podle \splittopskip.

Obdobně definujeme makra \doplnpravybox a \nactipravytext:

```
\newread\pravysoubor % zdrojový soubor s pravým textem
\newbox\pravybox % box, kam budeme načítat pravý text
\newdimen\prevdepthP % \prevdepth na konci boxu \pravybox
\let\nastavpravy=\relax % specifická nastavení pro pravý text
\def\doplnpravybox{\let\next=\relax
   \ifdim\ht\pravybox<1.25\vsize
      \ifeof\pravysoubor \else \nactipravytext
         \let\next=\doplnpravybox \fi\fi \next}
\def\nactipravytext{\let\soubor=\pravysoubor \kopiruj
   \setbox\pravybox=\vbox{\break \unvbox\pravybox
      \let\versnum=\versnumP \prevdepth=\prevdepthP
      \nastavpravy \input\jobname.tmp
      \global\prevdepthP=\prevdepth}%
   \setbox0=\vsplit\pravybox to 0pt }
```

Zde je kód makra \kopiruj spolu s kódem několika dalších maker, která s ním úzce souvisejí:

```
\newwrite\tempfile % pomocný soubor
\newcount\groupnum % počet otevřených skupin
\def\kopiruj{\begingroup \endlinechar=-1 \deaktivuj
   \aktivniznaky \nullfont \tracinglostchars=0
   \immediate\openout\tempfile=\jobname.tmp
   \let\next=\ctiradek \ctiradek
   \immediate\closeout\tempfile \endgroup}
\def\ctiradek{\read\soubor to\temp
   \immediate\write\tempfile{\temp}%
   \setbox0=\hbox{\temp\relax}%
   \ifx\temp\empty \ifnum\groupnum=0 \let\next=\relax \fi\fi
   \next}
\def\deaktivuj{\catcode`\ =12 \catcode`\$=12 \catcode`\&=12
  \catcode`\#=12 \catcode`\^=12 \catcode`\_=12 \catcode`\~=12 }
\begingroup
\catcode`\|=0 \catcode`\[=1  \catcode`\]=2 \catcode`\X=14
\catcode`\{=13 \catcode`\}=13 \catcode`\%=13 \catcode`\\=13
|gdef|aktivniznaky[X
   |catcode`{=13 |catcode`}=13 |catcode`%=13 |catcode`\=13
   |def{[|string{|ifhmode|global|advance|groupnum by 1 |fi]X
   |def}[|string}|ifhmode|global|advance|groupnum by -1 |fi]X
   |def%[|string%|ifhmode|expandafter|skryjkomentar|fi]X
   |def\[|string\|ifhmode|def|temp[]|expandafter|ctitoken|fi]]
|endgroup
```

```
\def\skryjkomentar#1\relax{}
\def\ctitoken#1{%
   \ifcat\noexpand#1a\edef\temp{\temp#1}\let\next=\ctitoken
      \else \ifx\temp\empty \def\temp{#1}\let\next=\relax
         \else \def\next{#1}\fi \testgroup \fi \next}
\def\idbgroup{bgroup} % identifikátor říd. sekvence \bgroup
\def\idegroup{egroup} % identifikátor říd. sekvence \egroup
\def\idbegingroup{begingroup} % ident. říd. sekv. \begingroup
\def\idendgroup{endgroup} % ident. říd. sekv. \endgroup
\def\testgroup{%
   \ifx\temp\idbgroup \global\advance\groupnum by 1 \fi
   \ifx\temp\idegroup \global\advance\groupnum by -1 \fi
   \ifx\temp\idbegingroup \global\advance\groupnum by 1 \fi
   \ifx\temp\idendgroup \global\advance\groupnum by -1 \fi}
```

Makro `\kopiruj` nejdříve pomocí záporné hodnoty registru `\endlinechar` sdělí input procesoru, aby na konec řádků nepřipojoval žádný znak. Dále znakům se speciálním významem přiřadí kategorii 12; výjimkou jsou kudrnaté závorky, zpětné lomítko a procento — ty se stanou aktivními znaky. Při zápisu do pomocného souboru (tj. mimo horizontální mód) expandují pouze na token kategorie 12. Ostatní jejich činnost bude probíhat v pracovním `\hboxu` (viz řádek 38): Levá kudrnatá závorka zvýší o jedničku hodnotu registru `\groupnum`, zatímco pravá kudrnatá závorka hodnotu registru `\groupnum` o jedničku sníží. Zpětné lomítko uloží identifikátor řídicí sekvence do těla definice makra `\temp` a poté prostřednictvím makra `\testgroup` rozhodne, zda se hodnota registru `\groupnum` zvýší, sníží, anebo zůstane beze změny. V makru `\testgroup` se testuje, zda identifikátor uložený do těla definice makra `\temp` je shodný s identifikátorem některé ze řídicích sekvencí `\bgroup`, `\egroup`, `\begingroup` a `\endgroup`. Podle potřeby je samozřejmě možno rozšířit makro `\testgroup` o další testy. Procento se postará o to, aby případné začátky a konce skupin, které se objeví v komentáři, neovlivnily hodnotu registru `\groupnum`.

Užití prázdného fontu `\nullfont`, který neobsahuje žádné znaky, mírně urychlí vytváření horizontálního seznamu uvnitř pracovního `\hboxu`; musíme ovšem zároveň nastavit `\tracinglostchars` na nulu, aby nebyla do souboru log vypisována hlášení o chybějících znacích, jinak by se naopak činnost TEXu zpomalila.[3]

Přečtení jednoho řádku ze zdrojového souboru, zapsání tohoto řádku do pomocného souboru a vypuštění téhož řádku do pracovního `\hboxu` bude v rámci

[3] V dnešní době jsou počítače natolik rychlé, že mírné zrychlení, jakého lze dosáhnout užitím prázdného fontu, se jeví jako takřka bezvýznamné. Pozorný čtenář si navíc mohl všimnout toho, že naše makra nejsou optimalizována z hlediska rychlosti zpracování, nýbrž že jsme se snažili o co možná nejlépe čitelný kód. Zde jsme ale chtěli poukázat na existenci primitivu `\nullfont` a na jeho možné využití v praxi.

makra `\ctiradek` opakováno pořád dokola. Tento cyklus bude ukončen tehdy, když bude ze zdrojového souboru přečten prázdný řádek a současně hodnota registru `\groupnum` bude rovna nule. V makru `\ctiradek` není test, zda bylo dosaženo konce zdrojového souboru. Přesto bude cyklus na konci zdrojového souboru řádně ukončen i v případě, že uživatel nenechá na konci zdrojového souboru prázdný řádek. TEX totiž automaticky připojuje prázdný řádek na konec každého souboru, který je otevřen pomocí příkazu `\openin`.

Nyní se dostáváme k problematice synchronizace obou textů. Nejdříve si nastíníme postup řešení této problematiky; k tomu nám pomůže následující úvaha. Otevřeme-li si bibli, již po zběžném zhlédnutí jedné strany poznáme, že text je rozčleněn do poměrně krátkých úseků, nazývaných „verše“.[4] Hned by nás mohlo napadnout, že při zrcadlovém vydání řeckého originálu a českého překladu Nového zákona by bylo vhodné, aby na každé dvojstraně byly stejné verše originálu i překladu. Tuto myšlenku rozvineme dále, neboť víme, že prakticky všechny řecké a latinské texty jsou členěny podobným způsobem a že většina překladatelů vyznačuje ve svých překladech členění korespondující s členěním originálu. Představme si tedy, že na začátku každého verše je pomocí příkazu `\mark` vložena značka, jejíž obsah udává pořadí tohoto verše v celém textu (průběžné číslování veršů je samozřejmě provedeno pro levý a pravý text zvlášť). Za takové situace můžeme pomocí příkazů `\vsplit to\vsize` a `\splitbotmark` zjistit, které verše se vejdou na levou stranu právě sázené dvojstrany a které verše na stranu pravou. Pokud by se ukázalo, že číslo posledního verše v materiálu určeném pro jednu stranu je vyšší než číslo posledního verše v materiálu určeném pro protilehlou stranu, můžeme odkrojený materiál, na jehož konci je verš s vyšším číslem, vrátit do příslušného `\vbox`u a z tohoto `\vbox`u odkrojit materiál o výšce menší než je `\vsize` tak, aby číslo posledního verše v tomto materiálu bylo shodné s číslem posledního verše v materiálu, který jsme odkrojili pro protilehlou stranu.

Implementace synchronizačních značek by mohla vypadat takto: Zařídíme, aby v souboru s levým i pravým textem byla na začátku každého verše zapsána řídicí sekvence `\vers`, a do souboru maker přidáme tyto řádky:

```
67 \newcount\versnumL \newcount\versnumP
68 \def\vers{\global\advance\versnum by 1
69    \leavevmode\mark{\the\versnum::}}
```

V definici makra `\vers` jsme využili toho, že se při načítání souborů mění význam řídicí sekvence `\versnum` podle toho, jestli se právě načítá levý, nebo pravý text (viz řádky 11 a 25).

Prolistujeme-li si více zrcadlových vydání řeckých a latinských textů a jejich překladů, zjistíme, že se často v záhlaví objevuje číslo (přesněji řečeno označení)

[4] Nenechme se zmást: Pojem „verš“ tu neznamená totéž, co se jako verš označuje v poezii. Dalo by se spíše říci, že verš zde znamená asi tolik co „věta“. Slůvko „asi“ je důležité, neboť oba pojmy — „verš“ a „věta“ — nelze plně ztotožnit.

prvního a posledního verše na aktuální straně. K tomuto účelu se v TEXu využívá značek typu mark. TEX přitom disponuje jedinou třídou těchto značek, a tu jsme právě vypotřebovali na synchronizační značky; je však jasné, že číslo vložené do synchronizační značky se vůbec nemusí shodovat s konvenčním označením verše. Navíc bychom pro sazbu záhlaví potřebovali znát nejen označení veršů, ale také označení kapitol apod. Do jednotlivých značek typu mark budeme tedy muset vměstnat více informací.

Prostor v každé značce typu mark rozdělíme na dva oddíly: První oddíl bude obsahovat synchronizační číslo získané z registru `\versnumL`, resp. `\versnumP`; druhý oddíl bude mít plně k dispozici uživatel. Pokud bude uživatel chtít do tohoto oddílu vložit nějakou informaci, místo makra `\vers` užije makra `\Vers`.

```
\def\Vers#1{\global\advance\versnum by 1
   \leavevmode\mark{\the\versnum:#1:}}
```

Následující makra nám zpřístupní informace ze značek typu mark:

```
\newcount\botversnumL
\def\readmarksL{%
   \expandafter\doreadfirstL\splitfirstmark
   \expandafter\doreadbotL\splitbotmark}
\def\doreadfirstL#1:#2:{\def\firstmarkL{#2}}
\def\doreadbotL#1:#2:{\botversnumL=#1 \def\botmarkL{#2}}
\newcount\botversnumP
\def\readmarksP{%
   \expandafter\doreadfirstP\splitfirstmark
   \expandafter\doreadbotP\splitbotmark}
\def\doreadfirstP#1:#2:{\def\firstmarkP{#2}}
\def\doreadbotP#1:#2:{\botversnumP=#1 \def\botmarkP{#2}}
```

Makro `\readmarksL` uloží do registru `\botversnumL` synchronizační číslo získané z poslední značky typu mark na levé straně; dále do těla definice maker `\firstmarkL` a `\botmarkL` uloží obsah uživatelského oddílu z první a poslední značky typu mark na levé straně. Obdobně funguje makro `\readmarksP`, které je ovšem určeno pro pravé strany. S makry `\firstmarkL`, `\botmarkL`, `\firstmarkP` a `\botmarkP` může uživatel pracovat ve výstupní rutině stejným způsobem, jako se běžně pracuje s primitivy `\firstmark` a `\botmark`.

V případě, že by se náhodou na nějaké straně neobjevila žádná značka typu mark, budeme pracovat s obsahem poslední značky na předcházející straně; k tomu slouží tyto definice:

```
\def\readlastpagemarkL{%
   \expandafter\doreadfirstL\lastpagemarkL
   \expandafter\doreadbotL\lastpagemarkL
   \let\lastsplitmarkL=\lastpagemarkL}
```

```
\def\readlastpagemarkP{%
   \expandafter\doreadfirstP\lastpagemarkP
   \expandafter\doreadbotP\lastpagemarkP
   \let\lastsplitmarkP=\lastpagemarkP}
```

Teď si ukážeme definice maker, která budou z boxů `\levybox` a `\pravybox` odkrajovat jednotlivé strany.

```
\newbox\levastrana
\newbox\zalohalevybox
\newdimen\vsizeL
\def\odkrojlevou{\setbox\zalohalevybox=\copy\levybox
   \ifdim\vsizeL>0pt
      \setbox\levastrana=\vsplit\levybox to\vsizeL
      \edef\lastsplitmarkL{\splitbotmark}%
      \ifx\lastsplitmarkL\empty \readlastpagemarkL
         \else \readmarksL \fi
   \else
      \setbox\levybox=\vbox{\line{}\nobreak\vskip-\topskip
         \nointerlineskip\line{}\unvbox\levybox}%
      \setbox\levastrana=\vsplit\levybox to-\topskip
      \readlastpagemarkL
   \fi}
\newbox\pravastrana
\newbox\zalohapravybox
\newdimen\vsizeP
\def\odkrojpravou{\setbox\zalohapravybox=\copy\pravybox
   \ifdim\vsizeP>0pt
      \setbox\pravastrana=\vsplit\pravybox to\vsizeP
      \edef\lastsplitmarkP{\splitbotmark}%
      \ifx\lastsplitmarkP\empty \readlastpagemarkP
         \else \readmarksP \fi
   \else
      \setbox\pravybox=\vbox{\line{}\nobreak\vskip-\topskip
         \nointerlineskip\line{}\unvbox\pravybox}%
      \setbox\pravastrana=\vsplit\pravybox to-\topskip
      \readlastpagemarkP
   \fi}
```

Předpokládejme, že výchozí hodnota `\vsizeL` i `\vsizeP` je rovna `\vsize`. Makra `\odkrojlevou` a `\odkrojpravou` odkrojí z boxů `\levybox` a `\pravybox` potenciální levou a pravou stranu, tyto dvě potenciální strany uloží do boxů `\levastrana` a `\pravastrana` a do registrů `\botversnumL` a `\botversnumP` uloží číslo posledního verše na potenciální levé, resp. pravé straně.

Po odkrojení obou potenciálních stran převezme řízení makro \balancuj, které by mělo fungovat zhruba takto: Pokud je číslo posledního verše v boxu \levastrana vyšší než číslo posledního verše v boxu \pravastrana, sníží velikost \vsizeL o výšku jednoho řádku, obnoví původní obsah boxu \levybox a vyvolá znovu makro \odkrojlevou. Pokud je číslo posledního verše v boxu \levastrana nižší než číslo posledního verše v boxu \pravastrana, sníží velikost \vsizeP o výšku jednoho řádku, obnoví původní obsah boxu \pravybox a vyvolá znovu makro \odkrojpravou. Pokud je číslo posledního verše v boxu \levastrana shodné s číslem posledního verše v boxu \pravastrana, vyvolá makro \tiskni, které v hlavním vertikálním módu pomocí \unvbox vloží obsah boxu \levastrana do vertikálního seznamu a pomocí \penalty-10000 vyvolá výstupní rutinu, pak provede totéž s obsahem boxu \pravastrana.

Uvažujme však tento příklad: Verš číslo 10 je velice krátký a po rozlomení levého textu do jednotlivých řádků leží na témže řádku začátek verše číslo 10 i začátek verše číslo 11; zároveň je tento řádek posledním řádkem v boxu \levastrana. Číslo posledního verše v boxu \pravastrana je 10. Makro \balancuj zmenší velikost \vsizeL o jeden řádek, takže po dalším zlomu, který provede makro \odkrojlevou, bude vlevo číslo posledního verše 9. Nyní tedy makro \balancuj zmenší velikost \vsizeP, a tak na právě sázené dvojstraně zůstane zbytečně nevyužito místo o velikosti jednoho řádku. Náš příklad by mohl být rozvinut ještě dále: Kratší by byly i jiné verše, takže by vlevo ležel začátek veršů číslo 8 a 9 na jednom řádku a vpravo by stejný jev nastal u veršů číslo 9 a 10. Za této situace by ještě více místa přišlo nazmar. Kdybychom příklad dovedli ad absurdum, nakonec by nám na právě sázené dvojstraně nezbyl vůbec žádný text. Z tohoto příkladu si vezmeme ponaučení, a do definice makra \balancuj zakomponujeme mechanismy, které podobným případům zabrání.

```
% test, zda se minule vešlo více vlevo než vpravo:
\newif\ifvetsiL
% test, zda se minule vešlo více vpravo než vlevo:
\newif\ifvetsiP
% test, zda se má skutečně balancovat:
\newif\ifbalancuj
\def\balancuj{\let\next=\tiskni
   \ifdim\vsizeL>0pt \else
      \ifnum\botversnumL>\botversnumP \balancujfalse \fi \fi
   \ifdim\vsizeP>0pt \else
      \ifnum\botversnumP>\botversnumL \balancujfalse \fi \fi
   \ifbalancuj
      \ifnum\botversnumL>\botversnumP \vetsiLtrue
         \ifvetsiP
            \advance\vsizeP by\baselineskip
```

```
            \setbox\pravastrana=\vbox{}% uvolnění paměti
            \setbox\pravybox=\box\zalohapravybox
            \def\next{\odkrojpravou \tiskni}%
         \else
            \advance\vsizeL by-\baselineskip
            \setbox\levastrana=\vbox{}% uvolnění paměti
            \setbox\levybox=\box\zalohalevybox
            \def\next{\odkrojlevou \balancuj}%
         \fi
      \else
         \ifnum\botversnumL<\botversnumP \vetsiPtrue
            \ifvetsiL
               \advance\vsizeL by\baselineskip
               \setbox\levastrana=\vbox{}% uvolnění paměti
               \setbox\levybox=\box\zalohalevybox
               \def\next{\odkrojlevou \tiskni}%
            \else
               \advance\vsizeP by-\baselineskip
               \setbox\pravastrana=\vbox{}% uvolnění paměti
               \setbox\pravybox=\box\zalohapravybox
               \def\next{\odkrojpravou \balancuj}%
   \fi\fi\fi\fi \next}
```

Nyní nastala vhodná chvíle, abychom se začali zabývat tím, jaká opatření je třeba učinit v souvislosti s inserty.

Výchozí hodnota registrů `\vsizeL` a `\vsizeP` by měla odpovídat maximální výšce textu, který se vejde na příslušnou stranu. Pokud bude text obsahovat poznámky a jiné inserty s koeficientem $f > 0$, tato maximální výška bude menší než `\vsize`. Její hodnotu získáme tak, že kopii obsahu boxu `\levybox`, resp. `\pravybox` předáme standardním způsobem speciální výstupní rutině, která bude mít za úkol změřit výšku boxu 255 a pak zlikvidovat veškerý materiál z boxu 255 a ze všech boxů, ve kterých bude uložen obsah insertů. Zbytek materiálu, který zůstane ve vertikálním seznamu hlavního vertikálního módu, zlikviduje jiná speciální výstupní rutina. Jestliže budou tyto speciální výstupní rutiny vyvolány při kladném `\holdinginserts`, nebudeme se muset starat o to, které boxy jsou vyhrazeny pro inserty, neboť všechny inserty zůstanou v boxu 255.

Obsah boxů `\levastrana` a `\pravastrana` se má za pomoci makra `\tiskni` proměnit ve dvě strany uložené do souboru dvi. Kdybychom v makru `\tiskni` nevěnovali insertům náležitou péči, mohli bychom se dočkat nepříjemných překvapení. Algoritmus uzavření strany může totiž některé inserty vrátit z aktuální strany do přípravné oblasti; obsah těchto insertů by se pak buď vytiskl úplně jinde, než bychom chtěli, anebo by se vůbec nevytiskl. Proto pokud budou

některé inserty vráceny, vyvoláme speciální výstupní rutinu, a to při kladném `\holdinginserts`, takže všechny inserty zůstanou v boxu 255. Tato výstupní rutina přemístí inserty z boxu 255 do smluveného boxu, odkud je pak vrátíme do boxu `\levybox`, resp. `\pravybox`. Zda algoritmus uzavření strany vrátil do přípravné oblasti nějaké inserty, se dozvíme dotazem na `\insertpenalties`. Tento registr má totiž dva různé významy: 1. V době činnosti algoritmu plnění strany se v registru `\insertpenalties` sčítají některé penalty související s inserty. 2. Algoritmus uzavření strany prostřednictvím registru `\insertpenalties` předá výstupní rutině informaci o počtu insertů vrácených do přípravné oblasti; jedná se o ty inserty, které byly v aktuální straně před místem zlomu a měly příznak „čekej“.

```
\newdimen\vyskatextu
\def\urcivyskuoutput{\setbox255=\vbox{\unvbox255}
   \global\vyskatextu=\ht255
   \global\output={\setbox0=\box255}}
\def\urcivyskuvlevo{\global\output={\urcivyskuoutput}
   \global\holdinginserts=1
   \unvcopy\levybox \break \vsizeL=\vyskatextu}
\def\urcivyskuvpravo{\global\output={\urcivyskuoutput}
      \global\holdinginserts=1
      \unvcopy\pravybox \break \vsizeP=\vyskatextu}
```

Při tvorbě kritického aparátu se obvykle preferuje sazba všech poznámek na příslušné straně do jediného odstavce, jak je to popsáno v Knuthově TEXbooku na stranách 398—400. Při tomto způsobu sazby poznámek se pracuje jen s odhadem výšky výsledného odstavce s poznámkami, takže se občas stává, že výška poznámek plus výška boxu 255 je o něco větší než `\vsize`, což se ve výstupní rutině projeví přetečeným `\vbox`em s tělem strany. Poněvadž při zjišťování maximální možné výšky textu každé strany voláme výstupní rutinu „nanečisto“, nabízí se vhodná příležitost popsaný nedostatek odstranit. V této výstupní rutině můžeme totiž zjistit badness boxu s tělem strany; pokud by se podle hodnoty badness ukázalo, že se jedná o přetečený box, dočasně bychom pro tuto stranu zmenšili `\vsize`. Pokusíme se tedy k příkazům `\urcivyskuvlevo` a `\urcivyskuvpravo` vytvořit alternativu s touto funkcí; bude-li si jí uživatel přát využít, pomocí příkazu `\let` změní implicitní význam zmíněných dvou příkazů za alternativní.

```
\newdimen\zalohavsize
\newcount\pagebadness
\def\urcivyskuoutputalt{\global\vsize=\zalohavsize
   \setbox255=\vbox{\unvbox255}
   \global\vyskatextu=\ht255
   \setbox0=\pagebody \global\pagebadness=\badness
   \global\output={\setbox0=\box255} \global\holdinginserts=1
```

```
176    \ifnum\insertpenalties>0 \null\break \fi}
177 \newdimen\VsizeL
178 \def\urcivyskuvlevoalt{\VsizeL=\vsize \let\strana=L
179    \loop
180       \global\output={\urcivyskuoutputalt}
181       \global\holdinginserts=0
182       \global\vsize=\VsizeL \unvcopy\levybox \break
183       \vsizeL=\vyskatextu
184       \ifdim\VsizeL>0pt \else \pagebadness=0 \fi
185    \ifnum\pagebadness=1000000 % overfull box
186       \advance\VsizeL by -0.5\baselineskip
187    \repeat}
188 \newdimen\VsizeP
189 \def\urcivyskuvpravoalt{\VsizeP=\vsize \let\strana=P
190    \loop
191       \global\output={\urcivyskuoutputalt}
192       \global\holdinginserts=0
193       \global\vsize=\VsizeP \unvcopy\pravybox \break
194       \vsizeP=\vyskatextu
195       \ifdim\VsizeP>0pt \else \pagebadness=0 \fi
196    \ifnum\pagebadness=1000000 % overfull box
197       \advance\VsizeP by -0.5\baselineskip
198    \repeat}
```

Speciální výstupní rutinu, kterou jsme nazvali `\urcivyskuoutputalt`, musíme vyvolat při nulovém `\holdinginserts`, poněvadž potřebujeme zjistit, zda odhad výšky odstavce s poznámkami byl tak nepřesný, že po zařazení tohoto odstavce do těla strany se tělo strany nevejde do `\vbox`u o výšce `\vsize`. Na řádku 174 předpokládáme, že makro `\pagebody` je synonymem pro vytvoření `\vbox`u obsahujícího tělo strany; hodnotu jeho badness si uschováme do registru `\pagebadness`. Někdy je při sestavování těla strany potřeba vědět, zda se jedná o levou, nebo pravou stranu; k tomu je určena řídicí sekvence `\strana`, která má při sestavování levé strany význam `L` a při sestavování pravé strany význam `P`. Povšimněme si, že registry `\VsizeL` a `\VsizeP` jsou určeny pro algoritmus stránkového zlomu a udávají cílovou výšku těla strany, zatímco registry `\vsizeL` a `\vsizeP` jsou určeny pro operaci `\vsplit` a udávají výšku textu, který má být zařazen do těla strany.

```
199 \newcount\zalohavbadness \newdimen\zalohavfuzz
200 \newcount\lastpenaltyL \newcount\lastpenaltyP
201 \newbox\boxins % sem budeme odchytávat vrácené inserty
202 \def\tiskni{\vbadness=\zalohavbadness \vfuzz=\zalohavfuzz
203    \tisknilevou \tiskni pravou}
```

```
\def\tisknilevou{\let\lastpagemarkL=\lastsplitmarkL
   \setbox\zalohalevybox=\vbox{} % uvolnění paměti
   \global\output={\tisknioutput} \global\holdinginserts=0
   \let\strana=L \global\vsize=\VsizeL
   \unvbox\levastrana \lastpenaltyL=\lastpenalty \break
   \doplnlevybox
   \ifvoid\boxins \else
      \pridejinserty\levybox\lastpenaltyL \fi}
\def\tiskni pravou{\let\lastpagemarkP=\lastsplitmarkP
   \setbox\zalohapravybox=\vbox{} % uvolnění paměti
   \global\output={\tisknioutput} \global\holdinginserts=0
   \let\strana=P \global\vsize=\VsizeP
   \unvbox\pravastrana \lastpenaltyP=\lastpenalty \break
   \doplnpravybox
   \ifvoid\boxins \else
      \pridejinserty\pravybox\lastpenaltyP \fi}
\def\tisknioutput{\global\vsize=\zalohavsize \usersoutput
   \ifnum\insertpenalties>0 \saveins \fi}
\def\saveins{\global\holdinginserts=1
   \global\output={\global\setbox\boxins=\vbox{
      \unvbox255 \setbox0=\lastbox \unskip}
      \global\output={\setbox0=\box255}}
   \null\break}
\def\pridejinserty#1#2{\setbox#1=\vbox{\nobreak\vskip\topskip
   \line{}\nobreak\vskip-\topskip \unvbox\boxins
   \ifnum #2=20000 \nobreak\vfill\Supereject \fi \unvbox#1}}
\def\Supereject{\par\penalty20000\penalty-20000 }
```

V makru `\tiskni` je zapotřebí nejprve obnovit původní hodnotu registrů `\vbadness` a `\vfuzz`; při určování maximální výšky a při odkrajování jednotlivých stran budou totiž nastaveny na takovou hodnotu, aby byla potlačena varování `Underfull` a `Overfull \vbox`. Pak je vytištěna levá strana; hlavní činnost při tom vykoná výstupní rutina `\tisknioutput`, která je vyvolána příkazem `\break` poté, co je vypuštěn v hlavním vertikálním módu obsah boxu `\levastrana`. Pokud budou nějaké inserty vráceny z aktuální strany, příkaz `\saveins` se postará o jejich přesunutí do boxu `\boxins`, odkud poputují na začátek boxu `\levybox`, aby mohly být příště zařazeny na levou stranu. Na řádku 224 odebereme prázdný box, vytvořený o dva řádky níže, a mezeru z `\topskip`, takže v boxu `\boxins` zůstanou jen inserty. Poněvadž nemůžeme zjistit skutečnou hodnotu penalty v místě zlomu, nebude správně fungovat příkaz `\supereject`; uživatel by místo něj měl používat nově definovaný příkaz `\Supereject`. Podobně jako levá strana je vytištěna i pravá strana.

Cyklus, při kterém budou vytištěny jednotlivé dvojstrany celého dokumentu, se bude jmenovat `\synchronizuj`; zde je jeho kód:

```
\def\synchronizuj{\vbadness=10000 \vfuzz=\maxdimen
   \urcivyskuvlevo \urcivyskuvpravo
   \balancujtrue \vetsiLfalse \vetsiPfalse
   \odkrojlevou \odkrojpravou \balancuj
   \ifvoid\levybox
      \ifvoid\pravybox \let\next=\relax
         \else \let\next=\dokoncipravou \fi
   \else
      \ifvoid\pravybox \let\next=\dokoncilevou
         \else \let\next=\synchronizuj \fi
   \fi \next}
```

Makra `\dokoncilevou` a `\dokoncipravou` se uplatní tehdy, když se jeden z boxů `\levybox` nebo `\pravybox` úplně vyprázdnil a již není čím ho doplnit, zatímco ve druhém z těchto boxů je ještě nějaký materiál. Tato makra pak zajišťují, aby se tento zbylý materiál vysázel pouze na levé, resp. pravé strany a aby protilehlé strany zůstaly prázdné.

```
\def\dokoncilevou{\vbadness=10000 \vfuzz=\maxdimen
   \urcivyskuvlevo \odkrojlevou
   \vbadness=\zalohavbadness \vfuzz=\zalohavfuzz
   \tisknilevou
   \global\output={\usersoutput}
   \let\strana=P \line{}\vfil\break % prázdná pravá strana
   \ifvoid\levybox \let\next=\relax
      \else \let\next=\dokoncilevou \fi
   \next}
\def\dokoncipravou{\vbadness=10000 \vfuzz=\maxdimen
   \urcivyskuvpravo \odkrojpravou
   \vbadness=\zalohavbadness \vfuzz=\zalohavfuzz
   \global\output={\usersoutput}
   \let\strana=L \line{}\vfil\break % prázdná levá strana
   \tisknipravou
   \ifvoid\pravybox \let\next=\relax
      \else \let\next=\dokoncipravou \fi
   \next}
```

Zrcadlovou sazbu bude uživatel zahajovat pomocí makra `\zrcadli`. Jako první parametr uvede jméno souboru s levým textem, jako druhý parametr zapíše jméno souboru s pravým textem. Oba parametry budou mít jako separátor jednu mezeru. Makro `\zrcadli` se nejprve postará o to, aby zrcadlová sazba

začínala na prázdné levé (sudé) straně. Potom provede výchozí nastavení některých registrů a předá řízení makru `\synchronizuj`; tím se zahájí cyklus, při kterém se vytisknou jednotlivé strany dokumentu.

```
\def\zrcadli #1 #2 {\begingroup
   \novalevastrana
   \openin\levysoubor=#1 \openin\pravysoubor=#2
   \zalohavsize=\vsize \VsizeL=\vsize \VsizeP=\vsize
   \zalohavbadness=\vbadness \zalohavfuzz=\vfuzz
   \maxdeadcycles=10000
   \splittopskip=\topskip \splitmaxdepth=\maxdepth
   \edef\usersoutput{\the\output}
   \global\versnumL=0 \global\versnumP=0
   \def\lastpagemarkL{0::} \def\lastpagemarkP{0::}
   \global\prevdepthL=-1000pt \global\prevdepthP=-1000pt
   \doplnlevybox \doplnpravybox
   \synchronizuj
   \global\output=\expandafter{\usersoutput}
   \global\holdinginserts=0
   \endgroup}
\def\novalevastrana{\supereject
   \ifodd\pageno\line{}\vfil\break\fi}
```

Pozorný čtenář si zajisté všiml, že na některých místech našeho kódu je globální přiřazení hodnot do některých registrů zdánlivě zbytečné. Toto globální přiřazování má ovšem svůj význam. Každá změna globální hodnoty registru na hodnotu lokální spotřebuje dvě slova z paměťového limitu „save size“. Proto střídáním globálního a lokálního přiřazování hodnoty jednomu a témuž registru můžeme tento limit snadno překročit. To je ostatně důvod, proč Knuth v plainu doporučuje užívat pomocné registry se sudými čísly a s číslem 255 pro lokální přiřazování a pomocné registry s lichými čísly (vyjma 255) pro globální přiřazování; toto doporučení bohužel nebývá občas uživateli respektováno.

V tuto chvíli je celý náš úkol vyřešen a nám už jen zbývá říci si pár slov o tom, jak budeme moci v roli uživatele pracovat se skupinou maker, která byla v tomto článku popsána. Veškerá tato makra uložíme třeba do souboru s názvem `zrcadlo.tex`.[5] Každý text, který budeme chtít užít při zrcadlové sazbě, budeme mít uložen v samostatném souboru a ve všech těchto souborech jednotlivé „verše“ opatříme smluvenou synchronizační značkou — může to být buď přímo řídicí sekvence `\vers`, resp. `\Vers`, anebo jiná řídicí sekvence, kterou posléze nadefinujeme například tak, že nejdříve vytiskne konvenční označení verše, které bude mít uvedeno jako svůj parametr, a pak zavolá makro `\vers` či `\Vers`.

[5] Čtenář nemusí tato makra opisovat, neboť jsou ke stažení na autorových internetových stránkách http://www.volny.cz/petr-brezina/.

Celou sazbu budeme řídit z hlavního souboru, jehož základní struktura bude vypadat takto:

- načtení souboru maker `zrcadlo.tex` pomocí příkazu `\input`,
- různá nastavení a uživatelovy definice,
- specifická nastavení pro levý a pravý text prostřednictvím řídicích sekvencí `\nastavlevy` a `\nastavpravy`,
- spuštění makra `\zrcadli`,
- ukončení činnosti TEXu pomocí příkazu `\bye`.

Tuto strukturu si samozřejmě podle potřeby upravíme. Například můžeme zrcadlit více dokumentů za sebou.

Pokud bychom chtěli zrcadlit dvojici textů, v nichž by nebyly vyznačeny jednotlivé „verše“, nejjednodušší by bylo nastavit `\everypar={\vers}`. To je ovšem možné pouze za předpokladu, že každému odstavci z jednoho textu odpovídá právě jeden odstavec v textu druhém. Nebudou-li odstavce příliš dlouhé, výsledek bude uspokojivý. Nicméně nejlépe by bylo, aby oba texty byly rozděleny na úseky („verše“) zahrnující zhruba dva až tři řádky tištěného textu.

Při ručním dolaďování zrcadlové sazby můžeme v obou textech využít příkazu `\vadjust{\break}` pro vynucení stránkového zlomu na příslušném místě uvnitř odstavce.

Kromě třídy insertů `\footins`, kterou zřejmě použijeme pro kritický aparát a pro překladatelské poznámky, plain deklaruje ještě třídu insertů `\topins`, určenou pro plovoucí obrázky a tabulky. Na uživatelské úrovni se s touto třídou insertů pracuje pomocí řídicích sekvencí `\midinsert`, `\topinsert` a `\pageinsert`. Jedna z těchto řídicích sekvencí nebude v zrcadlové sazbě správně fungovat. Pilný čtenář si může rozmyslet, která to je a proč.

Ještě bychom chtěli upozornit na to, že inserty a značky typu `\write` by v zrcadlených textech měly být s předcházející linkou nebo boxem pevně spojeny, tak aby mezi oběma elementy nemohlo dojít ke stránkovému zlomu.

Summary: Parallel typesetting

The article presents an efficient solution to the problem of typesetting two texts in parallel on facing pages in bilingual editions. The solution assumes that the two texts are saved separately in two files and that they are divided into small sections, just as the Bible is divided into verses. This division makes it possible to synchronize the texts automatically. Each of the two texts can have its own footnotes, illustrations and other insertions as if it were an ordinary document, but the texts are broken into individual pages simultaneously in such a way that each odd page contains the same sections as the corresponding even page. The presented macros are available on author's website http://www.volny.cz/petr-brezina/.