Jak jsem dělal knihu klikací

Petr Olšák

V tomto článku popíšu postupy a makra, pomocí nichž jsem převedl svou knihu *TEXbook naruby* [1] do klikací podoby formátu `pdf`. Použil jsem modifikaci TEXu nazvanou `tex2pdf` od pana Han The Thanha. Proto bude řeč hlavně o tomto programu. Každý uživatel Internetu a čtecího programu Acroreader se může podívat na výsledek mého snažení například na `ftp://math.feld.cvut.cz/pub/olsak/tbn`.

Český jazyk nám umožňuje pořadím slov nepatrně měnit význam řečeného. V názvu tohoto článku jsem použil spojení „knihu klikací" místo „klikací knihu". Tím jsem chtěl naznačit, že nejprve jsem vytvořil knihu a potom (jako vedlejší efekt) jsem se už danou knihu snažil převést do klikací podoby. Cílem díla tedy od začátku bylo vytvořit knihu, která má existovat především v *papírové podobě*, pokud možno kvalitně vysázená a svázaná. Vůbec jsem nechtěl dělat příruční nápovědu k TEXu do počítače. Celé dílo bylo tedy především podřízeno koncepci formátu knihy, nikoli filosofii elektronického hypertextového dokumentu. Kdybych měl dělat elektronickou nápovědu pro TEX, udělal bych ji zcela jinak. Jistě mnozí čtenáři vědí, že jsem s počítačovou myší zcela nesrostl, ba právě naopak. Takže asi bych elektronickou nápovědu nedělal vůbec.

Zdrojový text knihy jsem napsal s hojným počtem symbolických odkazů typu `\ref{značka}` podstatně dříve, než jsem začal vůbec koketovat s možností pořízení hypertextové verze. Jakým konkrétním dialektem byl text dokumentu napsán se může každý podívat — v elektronické distribuci knihy je ukázkový soubor `tabulky.tex` obsahující jednu kapitolu knihy. Při zavedení hypertextových odkazů tedy nebylo potřeba dělat vůbec žádnou změnu ve zdrojovém textu. Pozměnil jsem pouze pár maker pro formátování knihy a výsledkem byl klikací dokument, který obsahuje přes 10 tisíc hypertextových odkazů.

O programu `tex2pdf`

V [3] zveřejnil autor programu `tex2pdf` své dílo. Rozhodl jsem se tuto práci použít. Popíšu nejprve, jak vypadala instalace*). Balík `tex2pdf` je veřejně přístupný na `ftp://ftp.muni.cz/pub/tex/local/cstug/thanh/tex2pdf` a je snadno instalovatelný na UNIXových systémech, které používají TEX podle Berryho instalace `web2c`. V mém případě šlo o SUN OS 4.1.3 a `web2c` jsem tam měl, takže půda byla připravena. Stačilo postupovat podle návodu v `README`. Tam se například můžeme dočíst, že je nejprve potřeba překontrolovat, zda je přítomný `web2c/tex` společně s knihovnou `kpathsea-2.6`. To bohužel není zcela samozřejmé, protože Karl Berry nechal `web2c/tex` už dva roky netknutý se starou knihovnou `kpathsea-1.8`, která

*) Přiznám se, že mě nebaví číst recenze typu, jak jsem „To" vybalil z krabice, jak jsem „To" instaloval, jak jsem „To" rozchodil či nerozchodil apod. V tomto případě mi ale připadá popis instalace poměrně důležitý.

obsahuje chyby a se kterou nejde `tex2pdf` přeložit. Je tedy nejprve nutné postupovat podle záplatovacího souboru `web2c.kpathsea-2.6.help`, který je přítomen v archívech CTANu v místě vlastního `web2c` TEXu.

Pokud používáme originální Knuthův TEX a nechceme o něj přijít, pak překontrolujeme, zda máme v `/usr/local/bin` *kopie* binárních souborů `virtex` a `initex` a nikoli jen jejich linky ukazující do `web2c/tex`. V adresáři `web2c/tex` totiž vzniknou nové spustitelné programy `virtex` a `initex` (zahrnující vlastnosti výstupu do `pdf`) a původní programy nenávratně zmizí.

Před kompilací `tex2pdf` je rovněž nutné nejprve přeložit knihovny `zlib` a `libpng`. Přiznám se, že s tou druhou jsem měl jisté problémy, protože tam někdo nechal v `make` souboru místo povelu `ranlib` povel `echo`.

Konečně jsem pozměnil změnový soubor `tex1.ch` tak, aby výsledný TEX zapisoval do `logu` a dalších pracovních souborů `česky` a nikoli `^^e8sky`. Na odpovídající místo (podle pořadí sekcí) jsem proto přidal:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% [4.49] Make characters 128--255 printable
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
@x
  (k<" ")or(k>"~")
@y
  (k<" ")or(k=invalid_code)
@z
```

Vyhledávací místa (cesty) jsem nechal v původním stavu, tj. programy si najdou co potřebují v košatém stromu adresářů v `/usr/local/lib/texmf` podle dohodnutého schématu. Komu se to nehodí, přečte si dokumentaci ke `kpathsea` a upraví soubor `texmf.cnf`.

Nyní už stačí spustit podle návodu `install.sh`. Nemusíme se bát — nic se neinstaluje. Název tedy neberme vážně. V adresáři `web2c/tex` se provede změna změnových souborů a kompilují se tam nové programy `virtex` a `initex`. Vlastní instalace typu „napíšu install a pak program používám“ není zatím v této verzi distribuce zahrnuta. Možná je to dobře, protože každý se může sám rozhodnout, jak zařídí koexistenci nových programů s původními originálními Knuthovými. Já jsem například instalaci provedl takto:

```
# cd /usr/local/bin
# mv virtex virtex-knuth
# mv initex initex-knuth
# cd /usr/local/lib/texmf/ini
# mv tex.pool tex.pool-knuth
# mv csplain.fmt csplain.fmt-knuth
# cd /usr/src/kpathsea-2.6/web2c/tex
# cp virtex initex /usr/local/bin
# cp tex.pool /usr/local/lib/texmf/ini
# cd /usr/local/lib/texmf/ini
```

```
# initex csplain.ini
```

Vidíte, že jsem si nejprve udělal zálohy `*-knuth`, ke kterým se po dokončení práce s formátem `pdf` vrátím. Asi lepší řešení by bylo pozměnit názvy formátů (například `csplain-pdf.fmt`) a volat při požadavku na `pdf` výstup místo `csplain` `csplain-pdf`. Protože ale `pdf` varianta TEXu se od původní liší jen přidáním některých primitivů a bez dodatečných zásahů generuje běžné `dvi`, dá se nějakou dobu celkem dobře existovat i s tímto pozměněným TEXem.

V distribuci `tex2pdf` je názorná ukázka `example.tex`, ze které je celkem přehledně vidět použití jednotlivých nových primitivů. Když jsem si pak výsledek mohl vyzkoušet v Acroreaderu a viděl jsem, co to dělá, získal jsem první představu, jak formát `pdf` a implementace `tex2pdf` fungují. Přiznám se, že o `pdf` nemám žádné podrobnější informace (nevlastním žádnou dokumentaci), takže ono `example.tex` bylo pro mě doslova darem.

Při pokusu zpracovat ukázku `example.tex` celou (včetně obrázku na konci) se mi současná verze programu zbortila. To je ovšem možné přisoudit některým chybám v systémových knihovnách nebo v knihovnách funkcí na zpracování obrázků. Můj záměr udělat knihu klikací to neovlivnilo, protože nemám v knize jediný obrázek. Ptáte se proč tam nejsou obrázky? Protože kniha je manuálem k TEXu a nikoli obrázkovou publikací pro děti.

Použité fonty v knize

V knize jsem použil CS-fonty, protože text knihy je hlavně o plainu a csplainu. Druhý zmíněný formát tyto fonty implicitně používá. Zdálo se mi neférové použít jiné fonty než o kterých mluvím v ukázkách.

Formát `pdf` ovšem nepodporuje fonty ve formátu METAFONTu. Podporuje pouze fonty Type1 nebo TrueType. Poslední jmenovaný formát jsem okamžitě zavrhl a začal pátrat po řešení vedoucí k Type1 formátu fontů. Bohužel, CS-fonty v tomto formátu zatím nemáme. Přesto se nakonec ukázalo, že existuje řešení prostinké jak koloběžka. Na Síti jsou totiž volně dostupné tzv. BaKoMa Type1 fonty zahrnující jednak všechny Computer Modern fonty a jednak všechny DC fonty. Přitom `tex2pdf` pracuje při závěrečném přibalení fontů do výstupu se souborem `texpdf.map`, který má velmi podobné vlastnosti, jako soubor `psfonts.map`, jenž důvěrně známe z `dvips`. Například jsou zde informace o tom, jak naložit s fontem `cmmi10`:

```
cmmi10 cmmi10 4 <cmmi10.pfb CMmathit.enc
```

První slovo označuje název fontu, který je použit přímo v TEXu nebo na který odkazuje použitý virtuální font. Program `tex2pdf` totiž umí (při vytváření `pdf` výstupu) číst i virtuální fonty a provádí expanzi virtuálních skriptů až na úroveň „skutečně použitého“ fontu. Jeho název pak hledá v `texpdf.map`. Druhé slovo je názvem fontu ve světě PostScriptu. V případě BaKoMa fontů má `cmmi10.pfb` skutečně název `cmmi10`, takže toto slovo je shodné s prvním slovem. Za jiných okolností by tam bylo třeba napsáno `Palatino-BoldItalic` nebo něco podobného. Pak následuje tajemná konstanta `4`, jejíž význam jsem příliš nepochopil. Souvisí to nějak s různými variantami formátů fontů v `pdf`. Pokorně tu čtverku obkresluji a funguje to. Podstatně méně rád píšu čtverku do zkouškových zpráv při zkoušení studentů z lineární algebry. Právě nyní probíhá zkouškové období a tato nemilá povinnost mi nastala.

Za menšítkem je vlastní název souboru s fontem (`*.pfb`). Poslední slovo označuje název souboru, který popisuje kódování použitého fontu z pohledu TeXu v terminologii PostScriptu. Jedná se o známý kódovací vektor, který se používá i v `dvips`. Pokud je například na šesté pozici v tomto vektoru napsáno `/Sigma` a TeX sází fontem `cmmi10` šestou pozici, algoritmus vybere z fontu `cmmi10.pfb` znak `/Sigma` bez závislosti na tom, na jaké pozici je tento znak v PostScriptovém fontu zrovna implementován.

Dopišme do `texpdf.map` například řádky:

```
csr10  dcr10  4 <dcr10.pfb  il2.enc
cstt10 dctt10 4 <dctt10.pfb it2.enc
```

Tím jsme programu `tex2pdf` prozradili, odkud má brát jednotlivé znaky pro realizaci fontu `csr10` a `cstt10`. Kódovací vektory CS-fontů (soubory `il2.enc` a `it2.enc`) jsem převzal z distribuce balíku `a2ac` skoro beze změny. Přesněji, provedl jsem jen dvě úpravy: 1. V souboru `it2.enc` jsem zaměnil `/.notdef` na pozici 32 za `/visiblespace`, abych nepřišel o možnost tisku vaničky pro vyznačení mezery (␣). 2. Slova `/dcaron` a `/tcaron`, pro jejichž zavedení jsem měl dobré důvody (viz [2]), nejsou respektována v použitých fontech BaKoMa. Proto jsem je byl nucen nahradit slovy `/dquoteright` a `/tqouteright`.

Vidíme, že není potřeba použít žádné virtuální fonty. Za tuto jednoduchost ovšem platíme. Přišli jsme o znak promile, škrtátko pro polské l a hlavně o skupinu jedenácti znaků ze začátku kódové tabulky CM fontů, obsahující velká písmena řecké abecedy (Γ, Δ, atd.). Poslední problém lze vyřešit dodatečným zavedením fontu pro matematiku:

```
\font\cmtenrm=cmr10  \textfont0=\cmtenrm
\font\cmsevenrm=cmr7 \scriptfont0=\cmsevenrm
\font\cmfiverm=cmr5  \scriptscriptfont0=\cmfiverm
```

Font `\tenrm` zůstává zaveden jako `csr10`, takže v textovém režimu nepřicházíme o češtinu a v matematickém jsme ji nikdy neměli. Zůstává nám tedy nevyřešeno promile a škrtátko pro polské l. Nad tímto detailem jsem mávl rukou a vytvořil `pdf` formát knihy s vědomím, že u hesla `\promile` a `\l` v části B knihy budou na určitých místech bílá místa. Kdyby to náhodou někomu nedalo spát, je potřeba zdůraznit, že technické prostředky na řešení zmíněných dvou znaků existují. Pomocí virtuálního fontu sestavíme znak promile ze znaku procento a maličké nuly, která v DC fontech je. Pro škrtátko se odkážeme ve virtuálním skriptu například do fontu `cmr10`, takže se škrtátko vykreslí z fontu `cmr10.pfb`. Já osobně ale považuji `pdf` formát jen za klikací atraktivitu a ne za solidní formát určený k profesionálnímu tisku. Proto také oželím ztrátu kvality kresby některých akcentů způsobenou použitím kreseb DC fontů místo původních CS-fontů.

V souboru `texpdf.map` jsou již všechny fonty `cm*` zaneseny tak, že se čtou soubory `cm*.pfb` a odpovídající kódovací vektory. Bohužel, soubory s těmito kódovacími vektory (například `CMmathit.enc`) přibaleny nejsou. Povedlo se mi sice na CTANu soubory stejných názvů najít, ale tyto soubory používají na hojných místech zcela jiné názvy pro jednotlivé znaky, než používají BaKoMa fonty. Výsledkem potom je, že v matematických vzorcích máme až příliš mnoho nevytištěných znaků. Nezbylo mi tedy nic jiného, než vzít odpovídající BaKoMa fonty ve formátu `afm`, prohlídnout, jaké jsou tam použity názvy znaků a v kódovacích vektorech to opravit. V původním `texpdf.map` také nebylo rozlišeno kódování pro font `cmr10` a třeba `cmtt10`. V obou případech bylo použito `CMtext.enc`. Pak ale nevytisknete backslash z `cmtt10` ani kroucené závorky. Provedl jsem některé opravy a domluvím se s panem Thanhem, aby tuto „fontovou podporu pro BaKoMa fonty" zařadil do své distribuce programu `tex2pdf`.

Na závěr tohoto odstavce se zmíním ještě o dalším problému. V knize jsem použil font `logo10` pro logo METAFONT a dále balík fontů `bbold` pro ukázku zařazení nového matematického fontu do TeXu. Font `logo10` není do BaKoMa fontů zahrnut a z balíku `paradissa` mi tento font nějak nefungoval. Acroreader ho odmítal zobrazit. Fonty `bbold` v Type1 formátu pravděpodobně vůbec neexistují. Podruhé jsem mávl nad `pdf` formátem rukou a napsal jsem do zdrojového souboru `tbn.tex` náhradní a velmi nouzové řešení:

```
\let\mflogo=\rm
\font\bbtext=cmmib10
\let\bbex=\tenex
```

Původní struktura odkazů v knize

Kniha *TeXbook naruby* má obrovské množství odkazů nejrůznějších typů. Jejich převodem do hypertextové podoby vznikne poměrně pořádná klikací atrakce. Popíšu nejprve strukturu odkazů v takovém stavu, v jakém je použita v tištěné verzi knihy (tj. bez hyperlinků). Kdo má knihu před sebou, může tuto část číst rychleji, protože ví, o čem hovořím.

Obsah knihy je členěn na kapitoly. Každá kapitola má ještě několik sekcí. Takové členění je v části A knihy. Dále v části B knihy jsou dodatky. Některé krátké dodatky nejsou členěny vůbec a rozsáhlejší dodatky (slovník syntaktických pravidel a slovník primitivů a maker) je dále členěn na hesla řazená v abecedním pořadí.

V knize je velké množství ukázek maker ve verbatim prostředí. Tyto ukázky mají číslovány řádky a v textu se na některá čísla řádků odkazuji. Ukázky jsou při zpracování knihy automaticky extrahovány do sumárního souboru všech ukázek maker `tbn.mac`. V tomto souboru se odkazuje zpětně na strany, na kterých se ukázky v knize vyskytují.

V textu knihy se odkazuji na jednotlivé sekce v části A a používám hesla, která lze vyhledat v části B. Dále se na mnoha místech odkazuji na čísla stran. V části B je u každého hesla sumární odkaz na všechna čísla stran a čísla řádků v ukázkách, ve kterých bylo heslo v knize použito. Jakými makry jsem tohoto efektu docílil popisuji v sekci 2.4 knihy.

Konečně v rejstříku odkazujeme zpětně na čísla stran a stejně tak činíme v seznamu všech použitých příkladů v knize. Na seznam literatury odkazujeme v textu běžným číslem (například [1]) a v seznamu literatury je uveden zpětně seznam všech stránek v knize, kde byl titul zmíněn.

Knihu považuji svým způsobem též za ukázku, co všechno lze v oblasti křížových referencí naprogramovat makrojazykem TEXu. Při zpracování knihy se vytvářejí kromě `dvi` (nebo `pdf`) automaticky tři soubory. Již zmíněný soubor všech ukázek maker `tbn.mac`, dále soubor pro pozdější vygenerování obsahu `tbn.toc` a konečně soubor všech podkladů pro křížové reference `tbn.ref`. Tento soubor obsahuje kolem dvanácti tisíc řádků plných údajů o odkazech. Z toho je možné soudit, že různými křížovými referencemi všeho druhu je kniha skutečně naplněna až po okraj.

Teprve při čtvrtém zpracování knihy TEXem se všechny křížové reference použijí jak mají. Z důvodu šetření pamětí TEXu jsou totiž do `tbn.ref` zanášeny jen takové údaje, které skutečně potřebují být někde odkazovány. Probereme si jednotlivé průchody:

1. průchod: Soubor `tbn.ref` neexistuje. TEX se proto domnívá, že neexistuje žádné heslo v části B, na které je nutno odkazovat a z něhož je nutno odkazovat zpět na stránky, kde bylo heslo použito. Proto do `tbn.ref` nezapisuje výskyt žádného hesla, ale po přečtení části B tam zapíše informaci o existenci všech hesel.

2. průchod: Ze souboru `tbn.ref` si TEX přečte všechna použitá hesla. Nyní pro každé takové heslo zapíše to `tbn.ref` záznam o jeho poloze na stránce a případně na řádku v ukázce. Při tisku části B jsou ale zatím sumární informace s odkazy na stránky s výskyty odpovídajícího hesla prázdné. Proto část B vyjde na méně stránek, než odpovídá definitivnímu rozměru knihy.

3. průchod: Soubor `tbn.ref` už je zcela zaplněn, jen možná pro některá hesla nemá správný údaj o straně. Je to z toho důvodu, že v předchozím průchodu vyšla část B na méně stránek, než je definitivní rozměr knihy. Nyní už bude část B obsahovat plné zpětné odkazy a zaplní správný počet stránek. Tím se vytvoří definitivně `tbn.ref`, kde je vše zaneseno v pořádku.

4. průchod: Načte se správný `tbn.ref` a vytvoří se definitivní sazba knihy. Pomocí UNIXového `diff` lze ověřit, že nově vytvořený `tbn.ref` se s předchozí verzí z 3. průchodu neliší.

Pro zajímavost uvedu ještě velikosti jednotlivých elektronických formátů. Zdrojové soubory `*.tex` mají dohromady velikost 1,25 MB. Ty byly napsány lidskou rukou. Ostatní formáty jsou generovány automaticky. Soubor `tbn.dvi` má velikost 1,7 MB, což vzhledem k množství vstupních informací a vzhledem k tomu, že soubor obsahuje údaje o naprosto přesné sazbě textu, není nijak velký objem. Vidíme tedy, že formát `dvi` byl velmi dobře navržen (podrobněji se o něm můžete dozvědět přímo v knize, kde mimo jiné popisuji TEXovské formáty `dvi` a `tfm`). PostScriptový výstup, určený pro tisk, má pro srovnání 3,2 MB (fonty tam jsou v rozlišení 600 dpi bitmapové). Konečně formát `pdf` v nekomprimované podobě má přes 9 MB, což je nepoužitelná obluda. Proto jsem volil v `tex2pdf` kladný `\compresslevel` a dosáhl jsem výstupního `pdf` ve velikosti 4,8 MB. Bohužel komprimovaná data v `pdf` umí číst až Acroreader verze 3.0, takže pro uživatele verze 2 je `tbn.pdf` nečitelný.

Přidáváme hyperlinky

Hyperlink se v dokumentu skládá aspoň ze dvou částí: aktivní odkaz a cíl. Kliknutím na plochu aktivního odkazu se zobrazí na obrazovce ta část dokumentu, která obsahuje cíl. Na společný cíl může směrovat více aktivních odkazů. Všechny tyto aktivní odkazy a jejich společný cíl mají stejné přirozené číslo (tzv. číselný klíč), kterým jsou vzájemně propojeny. Toto číslo uživatel nevidí, ale při implementaci maker pro `pdf` formát nás bude toto číslo samozřejmě zajímat.

Aktivní odkaz je určen plochou obdélníkového tvaru umístěnou někde v dokumentu. Jakmile kurzor myši posuneme nad tuto plochu, změní se tvar kurzoru (například z packy se stane vztyčený prst). Navíc ještě může být toto místo ohraničeno viditelným rámečkem a text v něm (pro přehlednost) volíme obvykle v jiné barvě. O volbu odlišné barvy textu hyperlinku se ovšem musíme postarat sami — to není samozřejmá vlastnost hyperlinku.

Pro vymezení plochy aktivního odkazu jsou v `tex2pdf` použity značkovací primitivy `\pdfannotlink` a `\pdfendlink`. Jejich přítomnost v textu nemá žádný vliv na umístění ostatní sazby nebo velikosti boxů. První značka označuje začátek (levý okraj) aktivní plochy a druhá pravý okraj. Nad účařím bude plocha tak veliká, jako je výška boxu, ve kterém se obě značky vyskytují současně. Analogicky hloubka tohoto boxu určuje přesah aktivní plochy pod účaří. Tyto údaje lze též zadat explicitně pomocí klíčových slov `height` a `depth` u primitivu `\pdfannotlink`. V takovém případě se při výpočtu aktivní plochy ignorují rozměry boxu. Značky počátku a konce sice musí být ve společném boxu (jinak se `tex2pdf` tak prudce rozčílí, že ukončí celou činnost), ale pokud jsou dodatečně algoritmem řádkového zlomu od sebe odtrženy do více boxů, pak jsou značky začátku a konce automaticky přidány do míst roztržení. Tím je umožněno nechat aktivní celý víceřádkový úsek odstavce. To jsem ve své knize ale nikde nepoužil.

Cíl hyperlinku je dán pouze bezrozměrným místem v dokumentu. Rozlišujeme několik kategorií cílů: vertikální, horizontální a jiné. Kategorie se liší pouze v tom, jak se prohlížeč `pdf` vyrovná s celkovým umístěním strany v místě cíle po aktivaci hyperlinku. Buď bude vidět cíl společně s plnou šířkou strany, nebo bude vidět cíl společně s plnou výškou strany nebo nevím. Přiznám se, že jsem tyto vlastnosti zatím příliš neprokoukl a používám osvědčený `\pdfdestfith`*), což je zřejmě cíl, který ponechá velikost stránky na plnou její šířku.

Užitečné je vědět, že pokud existuje aktivní odkaz, který nemá odpovídající cíl, přesto se `pdf` vytvoří a uživatel Acroreaderu pak kliká naprázdno. Pokud zpracováváme pokusně jen část knihy, je tento jev velmi častý. Také je dobré vědět, že hyperlinky hned po prvním průchodu fungují dopředu i zpět, což ve srovnání s generováním tištěných odkazů nemá obdobu. V `pdf` formátu nevadí ani přítomnost osiřelých cílů, které nemají na sebe žádný odkaz.

---

*) V nové verzi `tex2pdf` je místo `\pdfdestfith` použit primitiv `\pdfdest` s různými parametry `fitph`, `fitpv`, `fitp`, `fitbh` atd.

Nejprve jsem navrhl makra, která tvoří rozhraní mezi makry již použitými v knize a novými primitivy z `tex2pdf`. Pokud nepoužiji výstup do `pdf`, chtěl bych, aby makra pracovala „naprázdno“, tj. třeba takto:

```
\def\beglink#1{}   % Začátek odkazu, #1 je identifikátor klíče
\def\endlink{}     % Konec odkazu
\def\aimlink#1{}   % Cíl odkazu, #1 je identifikátor klíče
\def\pglink#1{#1}  % \pglink{28} tiskne číslo 28 jako
                   % aktivní odkaz na stranu 28
```

Použití uvedených čtyř maker jsem zařadil do stávajících maker knihy na hojná místa a překontroloval, zda tyto zásahy nezměnily dosavadní zpracování knihy. Například pro tisk řádku obsahu s názvem sekce jsem pozměnil makro `\subtocline` následovně:

```
% v tbn.toc je třeba: \subtocline{1.3}{Token procesor}{19}{token}
\def\subtocline #1#2#3#4{%
   \line{\qquad #1\quad \hbox{\beglink{sec/#4}#2\endlink}%
       \puntiky\hbox to2em{\hfil\pglink{#3}}}\cvak}
```

V obsahu tedy budeme mít aktivní jednak názvy sekcí a jednak čísla stran. Názvy sekcí odkazují přímo na začátek sekce, zatímco čísla stran odkazují na začátek strany, na které sekce začíná.

V makru `\sub`, které je použito v místě začátku sekce, píšu:

```
% použití: \sub Token procesor [token]
\def\sub #1 [#2]{\par\goodbreak
   \advance\secnum by1
   \noindent{\bbf \the\chapnum.\the\secnum. #1}\par\nobreak
   \aimlink{sec/#2} % <<< --- toto souvisí s hypertextovým cílem
   ... zapiš odpovídající informace do tbn.toc a tbn.ref}
```

Z důvodu přehlednosti jsem si tuto ukázku lehce zestručnil a nerozepisuji, jak se zapisují informace do `tbn.toc` a `tbn.ref`. To není věc, kterou nyní sledujeme.

Nastal čas vytvořit klikací folklór. K tomu stačí předefinovat zmíněná makra například takto:

```
\newcount\numlink \newcount\sumlink  % Číslo PDF odkazu
\pdfoutput=1
\def\pdfsetcmykcolor#1{\special{PDF:#1 k}}
\def\CadetBlue{\pdfsetcmykcolor{0.62 0.57 0.23 0}}
\def\Black{\pdfsetcmykcolor{0 0 0 1}}  % Barva pro hyperlinky
\def\beglink#1{\convertkey{#1}\CadetBlue
   \pdfannotlink height8pt depth3pt
        attr{/Border [0 0 0]} \numlink \relax}
\def\endlink{\pdfendlink\Black}
\def\aimlink#1{\convertkey{#1}\pdfdestfith \numlink \relax}
\def\convertkey#1{{% Konvertuje textový klíč na číslo do \numlink
```

```
\expandafter\ifx \csname #1\endcsname \relax
   \global\advance\sumlink by1
   \expandafter\xdef \csname #1\endcsname{\the\sumlink}%
   \global\numlink=\sumlink
\else \global\numlink=\csname #1\endcsname\relax
\fi}}
```

Vidíme, že pomocí příslušného `\special` jsme schopni „namíchat jakoukoli barvu“ pro běžný text i pro text aktivních odkazů. Zde byla pro texty odkazů použita barva modrá, soudě podle názvu jde o barvu uniforem vojenského námořnictva či letectva. V distribuci `tex2pdf` je soubor `colorpdf.tex`, kde je předdefinováno velké množství různých barviček. Vidíme, že se barva deklaruje obvyklým způsobem po jednotlivých složkách v CMYK barevném plánu.

Parametrem `attr`, který přísluší primitivu `\pdfannotlink`, je možno dát formátu `pdf` vzkaz v jeho řeči o vzhledu aktivního odkazu. Já tu řeč moc neumím, pouze vím, že nenapíše-li se `/Border [0 0 0]`, budou mít aktivní odkazy černé rámečky, které v textu knihy poměrně dost ruší.

Těžiště našeho makra spočívá v pomocném makru `\convertkey`, které konvertuje textový klíč na přirozené číslo. Přitom tato konverze musí být jednoznačná. Například po `\convertkey{sec/token}` obdržíme v registru `\numlink` číslo (řekněme) 457 a příště, až se budeme znovu ptát na „`sec/token`“, dostaneme totéž číslo. Položíme-li dotaz s jiným identifikátorem klíče, vrátí nám makro do `\numlink` jiné číslo než 457. Primitivy implementující hyperlinky (`\pdfannotlink` a `\pdfdestfith`) vyžadují totiž ke své práci numerický klíč a ne mnemotechnické slovo*).

Zbývá nám ještě ukázat kód makra `\pglink`, které vysází svůj číselný parametr jako hypertextový odkaz na stránku odpovídajícího čísla.

```
\def\pglink#1{\hbox{\beglink{pg/#1}#1\endlink}}
\let\orishipout=\shipout
\def\shipout#1#2{\orishipout#1{\aimlink{pg/\the\pageno}#2}}
```

Zde jsme sáhli na primitiv `\shipout`, který jsme předefinovali tak, aby vkládal do výstupního seznamu i `\aimlink` s odpovídajícím číslem strany.

Zajímavé je též makro, které rozhoduje, zda vytištěná řídicí sekvence v ukázce maker „zmodrá“ (přesněji stane se aktivním odkazem), nebo zůstane černá. Kód tohoto makra zde nebudu uvádět, protože souvisí s dalšími záležitostmi a celé je to velmi rozsáhlé. Zmíním pouze princip tohoto makra.

Jak již bylo řečeno, při zpracování části B se pro každé heslo uloží do souboru `tbn.ref` informace o výskytu tohoto hesla. Při druhém zpracování si TEX z tohoto souboru přečte všechny názvy použitých hesel a udělá si jasno o tom, které heslo

*) Poznamenejme, že nová verze `tex2pdf`, kterou jsem ale při formátování knihy nepoužil, zahrnuje možnost čtení textových klíčů přímo na úrovni primitivů `\pdfannotlink`, `\pdfdest` a `\pdfoutline`. V takovém případě je zřejmě makro `\convertkey` zbytečné.

je v části B použito a které ne. Když pak například ve verbatim ukázce zpracovává heslo `\def`, ověří si, že toto heslo je v části B uvedeno a zpracuje je jako hyperlinkový odkaz. Když ale zpracovává třeba heslo `\memakro`, zjistí, že v části B nic takového není a ponechá text `\memakro` černý. Výsledkem je strakatá modročerná ukázka, kde vše, co je modré, je primitivem nebo makrem plainu vysvětleným v části B a vše, co je černé, je jen pomocným nebo nově definovaným makrem. V části B u jednotlivých hesel jsou zpětně odkazy na všechny výskyty hesla v knize a používají se tam další hesla vyhledatelná jinde v části B. Čtenář-uživatel se tedy může proklikat až do Bohnické léčebny.

Aby bylo takové zpracování modročerných ukázek možné, je samozřejmě v tomto „verbatim" režimu backslash aktivní. Aby to všechno fungovalo včetně extrahování ukázek do souboru `tbn.mac`, je to navíc poměrně komplikované. Kdo má zájem se do maker ponořit, může. Soubor maker `tbn.tex`, kterými byla kniha zpracována, je volně k dispozici v rámci elektronické distribuce knihy.

Přidáváme záložky dokumentu

Formát `pdf` má ještě jednu užitečnou vlastnost, kterou jsem nechtěl nechat stranou. Lze v něm deklarovat tzv. záložky. Jedná se o hypertextové odkazy, které nejsou umístěny v dokumentu, ale na okraji v samostatném rámečku. Tyto odkazy jsou tedy k dispozici pořád bez závislosti na zrovna zobrazené stránce dokumentu a váží se na cíl uvnitř dokumentu. Klikne-li uživatel na text záložky, zobrazí se mu stránka s odpovídajícím cílem.

Odkazy typu záložka lze navíc uspořádat v postranním rámečku do stromové struktury. Každá záložka může mít určité množství potomků (to jsou zase záložky) a navíc je možné předem definovat, zda implicitně uživatel ono potomstvo záložky vidí (otevřená záložka), nebo se mu objeví až po kliknutí na určitý symbol (zatím zavřená záložka).

Každého okamžitě napadne, že je nanejvýš žádoucí převést do záložek stromovou strukturu obsahu knihy. Tím má uživatel obsah neustále před sebou a nemusí se k němu vracet na konkrétní stránku dokumentu.

Záložky musí být vloženy v pořadí, v jakém se objeví v postranním rámečku a musí obsahovat text záložky a číselný klíč cíle (ono přirozené číslo, o kterém jsme už mluvili v souvislosti s hyperlinky). Navíc u záložky musí být řečeno, kolik má záložka potomků a zda se má zobrazit otevřená nebo zavřená. Protože musíme vložit počet potomků do záložky dříve, než projdeme všechny následující potomky v dokumentu, je zřejmé, že tuto vlastnost lze programovat pomocí aspoň dvou průchodů TeXu.

K implementaci stromové struktury záložek obsahu knihy ovšem nepotřebujeme zakládat nový pracovní soubor. V souboru s podklady pro obsah `tbn.toc` totiž už od druhého průchodu můžeme zjistit celou strukturu stromu záložek a pak už jen záložky vkládat. Záložky lze vkládat kdekoli v dokumentu (to nemá žádný vliv na jejich chování). Logicky ale budeme řešit záložku v době, kdy budeme vytvářet její cíl.

Rozhodl jsem se mezi záložky vložit též všechna hesla v části B. Budeme mít tedy čtyři druhy záložek. 1. Odkazující na kapitoly (budou mít potomky). 2. Odkazující na sekce (bez potomků). 3. Odkazující na dodatky (s výjimkou dvou dodatků s hesly bez potomků). 4. Odkazující na hesla v dodatcích (bez potomků). Všechny strukturované záložky budou implicitně zavřeny. Uživatel tedy nejprve uvidí jen názvy kapitol a dodatků. Klikne-li na příslušný symbol u názvu kapitoly, objeví se mu všechny sekce dané kapitoly. Klikne-li na vybranou sekci, může začít číst. Podobně po otevření záložky dodatku se všemi primitivy a makry se uživateli v prostoru záložek všechny tyto primitivy a makra objeví v abecedním pořádku. Je jich kolem tisíce. Pak uživateli stačí jen vybrat to heslo, které ho zajímá.

Nejprve jsem navrhl makro `\outlink`, které tvoří rozhraní mezi ostatními makry pro formátování knihy a použitými primitivy pro hyperlinky. Makro má tři parametry. Prvním parametrem je mnemotechnický klíč shodný s klíčem cíle, druhým parametrem je počet potomků (které budou následovat) a třetím parametrem je vlastní text záložky. Prázdná varianta makra (pro případ výstupu do `dvi` a nikoli do `pdf`) vypadá tedy takto:

```
\def\outlink #1#2#3{}
```

Varianta makra v případě výstupu do `pdf` je následující:

```
\def\outlink #1#2#3{{%
   \def\tt{}\def~{ }\def\char`{\string}\def\TeX##1{TeX}%
   \convertkey{#1}% klíč cíle
   \pdfoutline \numlink count -#2 % záložka je zavřená
       {#3}}} % vlastní text záložky
```

Za vysvětlení stojí zřejmě druhý řádek ukázky, kde jsou některé sekvence předefinovány. Do textu záložky se totiž dostane vše, i to, co bychom nechtěli. Například v nadpisu sekce používám zápis:

```
\sub Tabulky pomocí \tt \char`\\halign    [halign]
```

Protože jsem v makru `\sub` zařídil (zde neuvádím jak), že v okamžiku zápisu do `tbn.toc` mají sekvence `\\` a `\tt` význam `\relax` a sekvence `\char` je primitivní, zapíše se do `tbn.toc` skutečně text „`Tabulky pomocí \tt \char `\\halign`“. V době tisku obsahu se to správně přečte a vytiskne. Ovšem nic takového nechceme zapisovat do záložky. Proto jsou odpovídající sekvence na chvíli předefinovány. Náš příklad se tedy expanduje postupně na „`Tabulky pomocí \string\\halign`“ a do textu záložky se zapíše „`Tabulky pomocí \\halign`“. Protože je text záložky Acroreaderem interpretován jako string s céčkovou konvencí, změní se nakonec dvojitý backskash na jednoduchý.

Zabývejme se nyní výpočtem struktury záložek. Rozhodl jsem se pracovat s těmito řídicími sekvencemi: „`kap:`⟨*číslo kapitoly*⟩“ — expanduje na počet sekcí dané kapitoly. Dále „`app:`⟨*číslo dodatku*⟩“ — expanduje na počet hesel v daném dodatku.

Nejprve uvedu pomocná makra, která s těmito řídicími sekvencemi pracují:

```
\def\expnumber#1{\expandafter \ifx\csname#1\endcsname \relax 0%
```

```
    \else \csname#1\endcsname \fi}
\def\advancenumber#1{\tempnum=\expnumber{#1}\relax
    \advance\tempnum by1
    \expandafter\xdef\csname#1\endcsname{\the\tempnum}}
\def\extractkapnum#1.#2:{#1}
```

V souboru `tbn.toc` se vytvoří podklady pro obsah v tomto tvaru:

```
\kaptocline {1}{Vstupní části \TeX {}u}{10}
\subtocline {1.1}{Koncept vstupní brány \TeX {}u}{10}{koncept}
\subtocline {1.2}{Input procesor}{12}{input}
\subtocline {1.3}{Token procesor}{19}{token}
\kaptocline {2}{Expand procesor}{31}
\subtocline {2.1}{Definování maker}{31}{def}
...
```

Načtením takto členěného souboru `tbn.toc` získáme informace o počtech sekcí každé kapitoly:

```
\def\subtocline #1#2#3#4{%
  \advancenumber{kap:\extractkapnum#1:}%
  \line{... dále sázíme řádek obsahu stejně, jako prve ...}}
\kaptocline #1#2#3{\line{... sazba řádku obsahu beze změn ...}}
```

Podobně spočítáme počet hesel v části B při čtení souboru `tbn.ref`. Nyní stačí záložky prostě vložit na požadované místo. Třeba v definici `\kap` a `\sub` upravíme kód následujícím způsobem:

```
\def\kap#1 \par{\vfill\break \linenum=0 \secnum=0
   \global\advance\chapnum by1 \xdef\chapmark{\the\chapnum. #1}
   \headline={\hfil\starthead}
   {\bigbf \the\chapnum. #1}\par
   \aimlink{kp/\the\kapnum}
   \outlink{kp/\the\kapnum}{\expnumber{kap:\the\kapnum}}{#1}
   ... a dále zápis do tbn.toc}
\def\sub #1 [#2] \par{\par\goodbreak
   \advance\secnum by1
   \noindent{\bbf \the\kapnum.\the\secnum. #1}\par\nobreak
   \aimlink{sec/#2}
   \outlink{sec/#2}0{#1}
   ... a dále zápis do tbn.ref a tbn.toc}
```

Práce se záložkami má svá úskalí. Nelze vybrat font pro text záložek, protože tento font závisí na konfiguraci Acroreaderu v použitém operačním systému. Jsou-li tedy texty záložek české (v csplainu a v Unixovém TeXu je použito ISO-8859-2), pak se tyto texty správně zobrazí jen tehdy, pokud použitý operační systém pracuje s fonty ve stejném kódování a Acroreader je má nakonfigurovány pro zobrazování textů záložek. Z toho důvodu „Portable Document Format“ není zas tak úplně

„Portable", ale nad tím jsem jen mávl (do třetice) rukou. Mohl jsem sice vytvořit v TEXu makra, která by konvertovala názvy kapitol a sekcí na nehackovanou a necarkovanou variantu textu, ale nač si přidělávat zbytečnou práci, že?

Zde vidíme, že změna změnového souboru, zmíněná na začátku tohoto článku, má pro nás velký význam. Kdybychom ji neudělali, dostali bychom v českých textech do záložek `p^^f8^^ed^^b9ernou zm^^ec^^bb st^^f8^^ed^^b9ek`*).

Větvení výpočtu

V předchozích odstavcích jsme naznačili, že makra `\beglink`, `\endlink`, `\aimlink`, `\pglink` a `\outlink` necháme prázdná, pokud zpracováváme knihu do `dvi`. Naopak, nastavíme je na neprázdnou hodnotu, pokud generujeme `pdf`. K tomu se mi osvědčilo velmi prosté větvení:

```
% \let\pdfoutput=\undefined  %% Pokud chceme výstup jen do DVI
\ifx\pdfoutput\undefined
... definice prázdných maker:
    \beglink, \endlink, \aimlink, \pglink, \outlink
\else
\pdfoutput=1
\fontsubset=1
... definice hyperlinkových maker:
    \beglink, \endlink, \aimlink, \pglink, \outlink
... řešení náhradních fontů \fam0, \mflogo, \bbtext, \bbex
... zmenšení okrajů elektronického "papíru"
\fi
```

Jestliže nyní budeme zpracovávat knihu na Thanhově TEXu, bude se generovat `pdf` soubor. Pokud ji budeme zpracovávat klasickým TEXem, dostaneme staré známé `dvi`. Navíc, pokud chceme na Thanhově TEXu vytvořit výstupní `dvi`, stačí zrušit počáteční procento na prvním řádku ukázky.

Větvení jsem použil ještě v protitulu knihy. Tam mám v klasické variantě zrcadlový obrázek, zatímco v `pdf` variantě je vysvětlující text. K tomu stačilo psát:

```
%% Protitul
\ifx\pdfoutput\undefined
\input epsf
\vglue 7mm
\centerline{\epsfbox{protitul.eps}}
\else
\vglue 2cm
Toto je text knihy v~\uv{klikací} verzi ve formátu PDF.
Jsou zde použity náhradní fonty ve snížené kvalitě.
Proto nedoporučuji z~tohoto formátu tisknout.
Pro tisk použijte formát {\tt dvi} nebo
```

*) Zde bylo v ISO-8859-2 řečeno: „příšernou změť stříšek".

```
{\tt ps}, viz též soubor {\tt tbn-tisk.txt}.
\fi
\vfil\break
```

Závěrem

Chtěl bych poděkovat panu Thanhovi za jeho velmi dobře udělaný program `tex2pdf`. Hodně si cením i jeho ochoty pomoci mi v době, kdy jsem měl s tímto programem problémy. Pokud jsem v programu objevil nějakou závadu, měl jsem ji během pár hodin opravenu. Takový servis se hned tak nevidí.

Dále bych chtěl poznamenat, že mě ani po hraní s `pdf` formátem v Acroreaderu tato filosofie přenosu informací nenadchla a že zůstávám u starého známého tvrzení: *kniha je kniha.* S knihou si mohu vlézt do postele nebo najít jinou fyziologicky příjemnou polohu, zvolit osvětlení příjemné pro oči a plně a nerušeně se soustředit na informace v knize obsažené. Sebelepší počítač, sebelepší klikací vylomeniny a sebelepší obrazovka tuto pohodu nenahradí.

Tabulka použitých primitivů z `tex2pdf` implementace TEXu

Uvádím jen primitivy ze starší verze `tex2pdf`, které jsem použil. Program `tex2pdf` je stále ve vývoji a syntaxe některých primitivů už doznala změny. Proto doporučuji prostudovat nejprve dokumentaci k `tex2pdf` a `example.tex`.

Já jsem použil starší verzi `tex2pdf` hlavně proto, že jsem v ní měl už text knihy odladěn a nová verze (s bohatším jazykem primitivů a s novými klikacími možnostmi) byla v době závěrečného zpracování knihy teprve ve fázi testování. Až bude otestována, pak mi bude stačit během pár vteřin pozměnit použití primitivů v makrech `\beglink`, `\endlink`, `\aimlink` a `\outlink` a přechod na novou verzi bude proveden.

`\pdfoutput` — Je registr typu ⟨*number*⟩. Je-li kladný, bude výstup do `pdf`. Jinak do `dvi`. Implicitně 0. Záleží na jeho hodnotě při prvním `\shipout` v dokumentu.

`\pdfpagewidth` — Registr typu ⟨*dimen*⟩. Určuje šířku strany v `pdf` dokumentu. Není-li specifikován, bude počítán z registru `\hsize` plus okraj (1in+`\hoffset`) po obou stranách.

`\pdfpageheight` — Podobně jako `\pdfpagewidth`.

`\compresslevel` — Registr typu ⟨*number*⟩. Hodnota komprese textu ve výstupním `pdf`. Implicitně 0 (žádná komprese), maximálně 9 (největší komprese). Pozor, je-li `\compresslevel` > 0, je výstup čitelný jen v Acroreaderu verze $\geq$ `3.0`.

`\fontsubset` — Registr typu ⟨*number*⟩. Je-li kladný, budou do `pdf` zaváděny jen ty znaky fontů, které jsou použity. Nikoli tedy celé fonty.

`\pdfannottext width`⟨*dimen*⟩ `height`⟨*dimen*⟩ [`open`] `{`⟨*balanced text*⟩`}`
Vloží do dokumentu značku. Klikne-li uživatel na tuto značku,

otevře se text uvedený (po expanzi) v ⟨*balaced text*⟩. Je-li uvedeno nepovinné klíčové slovo `open`, inicializuje se značka jako otevřená.

`\pdfdestfith` ⟨*number*⟩
Cíl hyperlinku nebo záložky. Parametr ⟨*number*⟩ je klíč cíle.

`\pdfannotlink` [`height`⟨*dimen*⟩] [`depth`⟨*dimen*⟩] [`attr{`⟨*balanced text*⟩`}`] ⟨*number*⟩
Zahajovací značka odkazu hyperlinku. Parametr ⟨*number*⟩ je klíč shodný s klíčem cíle. Nepovinný ⟨*balanced text*⟩ udává parametry značky a `height` a `depth` její rozměry.

`\pdfendlink` Ukončující značka hyperlinku. Musí následovat ve stejném boxu, jako zahajující `\pdfannotlink`.

`\pdfoutline` ⟨*number*⟩ [`count`⟨*number*⟩] `{`⟨*balanced text*⟩`}`
Záložka. Povinný parametr ⟨*number*⟩ je klíč odpovídající klíči cíle a ⟨*balanced text*⟩ je (po expanzi) text záložky. Nepovinný `count`⟨*number*⟩ specifikuje počet potomků záložky, které budou za touto záložkou následovat. Implicitně 0.

`\pdfinfo` `{`⟨*balanced text*⟩`}`
Vloží do výstupního souboru údaje uložené v ⟨*balanced text*⟩. Jedná se o informace speciálních typů `/Author`, `/Title`, `/Subject`, `/Keywords`, `/CreationDate`, `/ModDate`, `/Creator` a `/Producer`, které se zobrazí při vyvolání „Document info“ ve čtecím programu.

`\special` `{PDF:`⟨*balanced text*⟩`}`
To je samozřejmě starý známý primitiv, ovšem za zmínku stojí, že při výstupu do `pdf` je `tex2pdf` schopen interpretovat jedině `\special` začínající na text „`PDF:`“. Jaké vzkazy se dají do výstupu prostřednictvím ⟨*balanced text*⟩ vkládat vyplývá zřejmě z dokumentace k `pdf`. Pro práci s obrázky ve formátu `png` je v `tex2pdf` implementován primitiv `\pdfimage`, který jsem ovšem v knize nepoužil.

Literatura

[1] Petr Olšák: *TEXbook naruby*, Praha 1996. Plný text knihy je k dispozici v elektronické podobě na `ftp://math.feld.cvut.cz/pub/olsak/tbn`. 468 stran.
[2] Petr Olšák: Program `a2ac` – manipulace s fontem na úrovni PostScriptu. *Zpravodaj CSTUGu*, 1/96, strany 23–30.
[3] Han The Thanh: Alternativní výstup programu TEX – PDF. *Zpravodaj CSTUGu*, 1/96, strany 69–85.