Zpravodaj Československého sdružení uživatelů TeXu

Petr Olšák

PDFuni - akcenty v PDF záložkách

Zpravodaj Československého sdružení uživatelů TeXu, Vol. 23 (2013), No. 1, 47–56

Persistent URL: http://dml.cz/dmlcz/150079

PDFuni – akcenty v PDF záložkách

Petr Olšák

Záložky v PDF dokumentu (outlines, bookmarks) jsou texty, které za jistých okolností zobrazují PDF prohlížeče a které jsou klikací: rozvírají se a nabízejí stromovou strukturu. A hlavně klik na text v záložce způsobí skok na tomu odpovídající místo v dokumentu. Záložky se nikdy netisknou. Mnozí uživatelé pdfTEXu si jistě všimli, že český nebo slovenský text má někdy sklon ke znehodnocení akcentovaných znaků, pokud jej posíláme do záložek. V tomto článku si vysvětlíme pozadí těchto problémů a popíšeme makro PDFuni pro plainTEX, které problém s českými a slovenskými texty v záložkách a dalších podobných místech řeší.

Klíčová slova

PDF, outlines, kódování, TEX, balíček maker, PDFuni

1. Kódovací standardy pro PDF stringy

PDF stringy jsou jednak texty v záložkách a jednak texty v informaci o PDF dokumentu vložené příkazem `\pdfinfo` o autorovi, názvu díla, klíčových slovech a podobně. Údaje z `\pdfinfo` lze z dokumentu získat programem `pdfinfo` na příkazovém řádku a některé PDF prohlížeče je také dokáží zobrazit.

PDF stringy nejsou součástí hlavního textu dokumentu, takže pro ně není použit font vybraný autorem dokumentu. Místo toho si PDF prohlížeč k zobrazení PDF stringů zavolá font, který je zrovna v době prohlížení po ruce a který mu nabídne operační systém.

Specifikace PDF formátu [1] vymezuje (v dodatku D) pro PDF stringy dvě možná kódování. První je jednobytové (tj. jeden znak je reprezentován jedním bytem) a jmenuje se `PDFDocEncoding`. Toto kódování se shoduje s viditelnými znaky z ISO-8859-1, takže Západoevropané s ním nemají problémy. Nevýhodou tohoto kódování je, že v něm nenajdeme většinu akcentovaných znaků české a slovenské abecedy. Nestaráme-li se o nic jiného, je uložený string interpretován PDF prohlížečem jako kódovaný podle `PDFDocEncoding`, což nám zákonitě poničí české texty.

Kvůli tomu jsem se rozhodl do makra OPmac [2] zařadit konverzní proces, který veškeré české a slovenské akcentované znaky převede do jejich ASCII podobných znaků bez akcentu a teprve takto převedený text uloží jako PDF string. V záložkách pak sice nevidíme akcenty, ale nevidíme tam ani nesmysly, ke kterým jsou záložky náchylné. I kdyby byly PDF stringy správně kódované pro češtinu (viz dále), může se stát, že se v prohlížeči nezobrazí správně, protože správné zobrazení závisí navíc na správném nastavení locales a na dalších systémových

doi: 10.5300/2013-1/47

parametrech. Rozhodl jsem se tedy pro robustní řešení: PDF stringy generované makrem OPmac budou jenom v ASCII. A nehodlám to měnit.

Nicméně uživatelé například balíčku `hyperref` si jistě všimli, že za jistých příznivých okolností (tj. `hyperref` pracuje v UNICODE a PDF prohlížeč má správně nastaveny veškeré systémové parametry) dokáží vygenerovat korektní české texty i v záložkách. Je to tím, že PDF specifikace nabízí druhou (a poslední) možnost kódování PDF stringů, kterou nazývá `UTF-16BE Unicode` (dále jen UNICODE). Kódování PDF stringu v tomto režimu se pozná tak, že první dva byty stringu mají hodnotu 254, 255. Je to prefix pro přepnutí do tohoto kódování. Podrobněji viz kapitolu 3.8.1 (String Types) v manuálu [1].

Nechceme-li ukládat do PDF stringu jednotlivé byty nativně, máme možnost je popsat pomocí backslashe následovaném třemi oktalovými číslicemi vyjadřujícími hodnotu bytu. String „Cvičení je zátěž“ tedy může v PDF stringu být zapsaný takto:

```
\376\377\000C\000v\000i\001\015\000e\000n\000\355\000
\040\000j\000e\000\040\000z\000\341\000t\001\033\001\176
```

V uvedeném příkladu vidíme nejprve byty 254, 255 zapsané oktalově, dále je oktalově zapsaná nula následovaná písmenem C (tyto dva byty reprezentují písmeno C v UTF-16 kódování). Podobně jistě dokážete dešifrovat písmena „v“ a „i“. Dále `\001\015` je 01,0D hexadecimálně, což reprezentuje znak „č“ v UNICODE. Dál si ten string můžete dočíst sami...

Právě takové stringy musíme generovat TEXem, pokud chceme vidět v záložkách správně napsané české a slovenské znaky. Je asi nemyslitelné, abychom psali ručně do dokumentu něco jako:

```
\pdfoutline goto name{cviceni} count0 {% záložka
\string\376\string\377\string\000C\string\000v\string\000i%
\string\001\string\015\string\000e\string\000n\string\000%
\string\355\string\000\string\040\string\000j\string\000e%
\string\000\string\040\string\000z\string\000\string\341%
\string\000t\string\001\string\033\string\001\string\176}%
\pdfdest name{cviceni} xyz\relax % cíl odkazu
```

Sice je tímto způsobem na primitivní úrovni vytvořena záložka s textem „Cvičení je zátěž“, ale kdybychom to takto dělali pořád, byla by to opravdu zátěž. Proto jsem vytvořil makro PDFuni, které uživatelům $\mathcal{C}\mathcal{S}$plainu ulehčí práci. Uživatelé LaTeXu nic takového nepotřebují, protože mají balíček `hyperref`.

2. Použití balíčku PDFuni

Protože OPmac generuje do záložek texty bez akcentů, je třeba zavolat PDFuni až po `\input opmac`. Balíček PDFuni předefinuje implicitní konvertor z OPmac pro výstup do PDF stringů (konvertor do ASCII) na nový konvertor do UNICODE. Takže stačí na začátku dokumentu psát:

```
\input opmac
\input pdfuni
```

Od této chvíle příkaz `\outlines`⟨*num*⟩ generuje do záložek správně české a slovenské texty. Využívá přitom automaticky generovaný obsah. OPmac nabízí ještě příkaz `\insertoutline{`⟨*text*⟩`}`, který vloží do záložek jeden údaj. Tento příkaz vkládá ⟨*text*⟩ bez konverze, z čehož plyne, že tam české akcenty nebudou jednoduše fungovat. Proto je potřeba použít příkaz `\pdfunidef\makro{`⟨*text*⟩`}`, který je definován v balíčku PDFuni a který vloží do `\makra` zkonvertovaný ⟨*text*⟩ v kódování UNICODE s oktalovými přepisy. Kategorie backslashe tam je 12 (obyčejný znak). Takže je potřeba psát:

```
\pdfunidef\tmp{Tady je český text}
\insertoutline{\tmp}
```

PDF dokument může obsahovat PDF stringy různě kódované. Takže funguje například i toto:

```
    % tento PDF string je v UNICODE:
\pdfunidef\tmp{Čeština je dřina}\insertoutline{\tmp}
    % tento PDF string je v PDFDocEncoding:
\insertoutline{Jan Hus vnesl do jazyka velkou hloupost:
               zavedl akcenty.}
```

Příkaz pdfTEXu `\pdfinfo` zanáší do PDF dokumentu doplňující informace, například takto:

```
\pdfinfo {/Author (Autor)
          /Title (Titul)
          /Creator (csplain + OPmac)
          /Subject (clanek)
          /Keywords (klicova slova)}
```

Je nutné vědět, že PDF stringem je zde každý údaj uzavřený do závorky, přičemž některé tyto PDF stringy mohou být kódovány podle PDFDocEncoding a jiné podle UNICODE. Údaje je tedy možno zanést třeba takto:

```
\def\author{Petr Olšák}\pdfunidef\author{\author}
\def\title{PDFuni - akcenty v PDF záložkách}\pdfunidef\title{\title}
\def\subject{článek}\pdfunidef\subject{\subject}
\pdfinfo {/Author (\author)
          /Title (\title)
          /Creator (csplain + OPmac)
          /CreationDate (D:20130107000000)
          /Subject (\subject)
          /Keywords (TeX; pdfTeX; PDF; OPmac)}
```

Poznamenejme, že další informace `/Producer` a `/ModDate` vloží většinou nejlépe pdfTEX sám, nicméně uživatel je rovněž může předefinovat. Není-li ani jeden údaj `/CreationDate` a `/ModDate` vyplněn, oba obsahují čas vzniku PDF souboru. Můžete si všimnout, že chcete-li `/CreationDate` vyplnit podle svého, je nutné dodržet formát `D:YYYYMMDDhhmmss`.

3. Možnost rozšíření kódovací tabulky o další znaky

Příkaz `\pdfunidef` konvertuje správně všechny viditelné ASCII znaky s kategoriemi 11, 12 (písmena, znaky) a dále znaky s kategoriemi 7, 8 (^, _). Ostatní viditelné ASCII znaky jiných kategorií (např. `$`, `&`) jsou ignorovány. Chcete-li do PDF stringu dopravit znak jiné kategorie než 7, 8, 11, 12, musí mít před sebou backslash, tedy `\\`, `\{`, `\}`, `\$`, `\&`, `\#`, `\~`, `\%`.

Dále `\pdfunidef` konvertuje znaky kategorie 11 nebo 12, které nejsou ASCII, pokud jsou zahrnuty do seznamu `\pdfunichars`. Tento seznam implicitně obsahuje znaky Áá Ää Čč Ďď Éé Ěě Íí ĹĺĽľ Ňň Óó Öö Ôô Řř Řř Šš Ťť Úú Ůů Üü Ýý Žž. Oktalový zápis těchto znaků (tedy výstup konverze) ve stejném pořadí je uložen v seznamu `\pdfunicodes`. Kódy jsou tam zapisovány bez backslashů a bez první nuly a jsou od sebe odděleny příkazem `\or`. Podívejte se do souboru `pdfuni.tex` nebo do následující sekce, jak vypadají implicitní hodnoty těchto seznamů. Chcete-li rozšířit tuto konverzní tabulku, je možné použít příkaz `\addto` z OPmac například takto:

```
\addto\pdfunichars{Ëë}  \addto\pdfinicodes{\or00313\or00353}
```

Uvedený příklad rozšiřuje seznam `\pdfunichars` o znaky Ë, ë a dále rozšiřuje seznam `\pdfunicodes` o kódy `\000\313` a `\000\353`. Na tyto kódy se budou znaky Ë, ë (v tomto pořadí) konvertovat.[1]

Příkaz `\pdfunidef` ještě před začátkem konverze provede úplnou expanzi svého parametru ⟨*text*⟩. Před touto expanzí je vhodné předefinovat některá makra, například `\def\TeX{TeX}`. Taková činnost je zanesena do seznamu

[1] Aby uvedený příklad fungoval, musejí mít znaky Ëë svou reprezentaci ve vnitřním kódování TEXu. To je u 16bitového TEXu zařízeno automaticky. Pro $\mathcal{CS}$plainu a encTEXu je možné použít třeba `\input t1code`.

`\pdfunipre`, který je (uvnitř skupiny) předřazen před expanzí parametru ⟨*text*⟩. Kromě speciálních expanzí maker tam jsou povelem `\let\makro\relax` zabezpečena makra, která expandovat v dané chvíli nechceme. Jsou to makra `\ae`, `\P` atd., která posléze vstoupí do konverze a promění se na svůj oktalový kód. Přidělení tohoto kódu se děje v seznamu `\pdfunipost` pomocí příkazu `\odef\makro`⟨*6ciferný kód*⟩⟨*mezera*⟩. Například příkazem `\odef\ae000346` říkáme, že makro `\ae` se zkonvertuje na `\000\346`. Čtenáři doporučuji podívat se na obsahy seznamů `\pdfunipre` a `\pdfunipost` do souboru `pdfuni.tex` nebo do následující sekce.

Chcete-li rozšířit konverzní tabulku týkající se maker, lze to udělat pomocí `\addto` například takto:

```
\addto\pdfunipre{\let\endash\relax}
\addto\pdfunipost{\odef\endash040023 }
```

Tento příklad přidává do tabulky údaj o konverzi makra `\endash`. Ve fázi expandování parametru je jeho expanze potlačena a dále se zkonvertuje na kód `\040\023`, což je dle UNICODE pomlka na půlčtverčík.

Kontrolní sekvence, které nejsou během expanze parametru ⟨*text*⟩ expandovány a nemají deklarován svůj kód pomocí `\odef`, jsou při konverzi ignorovány.

Nyní už čtenář patrně ví, proč při použití $\mathcal{CS}$plainu s encTEXem funguje správně i právnický znak §, ačkoli není uveden v seznamu `\pdfunichars`. EncTEX jej totiž mapuje na kontrolní sekvenci `\S`, takže kdykoli napíšeme §, promění se to automaticky v `\S`. Znak `\S` je ošetřen v seznamu `\pdfunipre`, kde je napsáno `\let\S\relax`. A v seznamu `\pdfunipost` je definice `\odef\S000247`.

4. Popis interních maker a s nimi spojených triků

Makra z `pdfuni.tex` obsahují několik zajímavých triků. Jejich popis je určen pro pokročilé TEXisty.

Soubor `pdfuni.tex` je kódován v UTF-8 kódování. Je nutné, aby byl přečten encTEXem nebo 16bitovým TEXem, protože chceme s jednotlivými akcentovanými znaky v TEXu pracovat jako s jedním tokenem. Proto je na začátku souboru test, zda je použit správný TEXový překladač:

```
\def\tmp#1#2\end{\if$#2$\else
\errmessage {This file is UTF-8 encoded. Use TeX+encTeX
             or 16bit TeX engine.}\fi}%
\tmp č\end
```

Dále na začátku souboru definujeme makro `\Bslash`, které expanduje na backslash kategorie 12. Totéž dělá OPmac s makrem `\bslash`, ale OPmac hodnotu tohoto makra v různých místech mění. Zde potřebujeme mít jistotu, že to bude pořád backslash. Kromě toho potřebujeme pracovat s pomocným čítačem

`\tmpnum`. Protože nevíme, zda je zavolán OPmac (makro PDFuni funguje i bez něj), je za předpokladu nedefinovanosti `\newcount` tento čítač deklarován.

```
\ifx\tmpnum\undefined \csname newcount\endcsname\tmpnum \fi
{\lccode`\?=`\\  \lowercase{\gdef\Bslash{?}}}
```

Následuje výpis implicitní hodnoty seznamů `\pdfunichars` a `\pdfunicodes`:

```
\edef\pdfunichars{\Bslash()
   ÁáÄäČčĎď ÉéĚěÍíĹĺĽľ ŇňÓóÖö ŐőŘřŔŕŠš ŤťÚúŮůÜü ÝýŽž
}
\def\pdfunicodes {\or00134\or00050\or00051\or             % \ ()
   00301\or00341\or00304\or00344\or01014\or01015\or       % ÁáÄäČč
   01016\or01017\or00311\or00351\or01032\or01033\or       % ĎďĚéĚě
   00315\or00355\or01071\or01072\or01075\or01076\or       % ÍíĹĺĽľ
   01107\or01110\or00323\or00363\or00326\or00366\or       % ŇňÓóÖö
   00324\or00364\or01124\or01125\or01130\or01131\or       % ŐőŘřŔŕ
   01140\or01141\or01144\or01145\or00332\or00372\or       % ŠšŤťÚú
   01156\or01157\or00334\or00374\or00335\or00375\or       % ŮůÜüÝý
   01175\or01176%                                         % Žž
}
```

Je vidět, že do seznamu jsou zařazeny i hodnoty `\,` `(`, `)`. Je to tím, že tyto znaky nechceme konvertovat do podoby `\000`⟨*znak*⟩, protože by v PDF stringu způsobovaly problémy (backslash je escape prefix a kulaté závorky ohraničují string). Je tedy bezpečnější je do PDF stringu zapsat v úplné oktalové notaci.

Pokračuje výpis implicitní hodnoty `\pdfunipre` a `\pdfunipost` a některých speciálních kódů.

```
\def\pdfunipre {\def\TeX{TeX}\def\LaTeX{LaTeX}\def~{ }\def\ { }%
   \let\ss\relax \let\l\relax \let\L\relax
   \let\ae\relax \let\oe\relax \let\AE\relax \let\OE\relax
   \let\o\relax \let\O\relax \let\i\relax \let\j\relax \let\aa\relax
   \let\AA\relax \let\S\relax \let\P\relax \let\copyright\relax
   \let\dots\relax \let\dag\relax \let\ddag\relax
   \let\clqq\relax \let\crqq\relax \let\flqq\relax \let\frqq\relax
   \let\promile\relax \let\euro\relax
   \def\\{\Bslash}\let\{\relax \let\}\relax
   \def\${\string$}\def\&{\string&}\def\_{\string_}\def\~{\string~}%
}
\def\pdfunipost {\odef\ss000337 \odef\l001102 \odef\L001101
   \odef\AE000306 \odef\ae000346 \odef\OE001122 \odef\oe001123
   \odef\o000370 \odef\O000330 \odef\aa000345 \odef\AA000305
   \odef\i001061 \odef\j002067  \odef\S000247 \odef\P000266
   \odef\copyright000251 \odef\dots040046 \odef\dag040040
   \odef\ddag040041 \odef\clqq040033 \odef\crqq040034
   \odef\flqq000253 \odef\frqq000273 \odef\promile040060
```

```
   \odef\euro040254 \odef\{000173 \odef\}000175 \odef\#000043
}
\def\pdfinitcode{\Bslash376\Bslash377}
\def\pdfspacecode{\Bslash000\Bslash040}
\def\pdfcircumflexcode{\Bslash000\Bslash136}
\def\pdflowlinecode{\Bslash000\Bslash137}
```

Nyní přistupme k tomu hlavnímu, k definici makra `\pdfunidef`⟨*makro*⟩`{`⟨*text*⟩`}`.

```
\def\pdfunidef#1#2{\bgroup \def\tmpa{\noexpand#1}%
   \pdfunipre \edef\tmp{#2}\edef\out{\pdfinitcode}%
   \def\odef##1##2##3##4##5##6##7
      {\def##1{&\edef\out{\out\Bslash##2##3##4\Bslash##5##6##7}}}%
            \pdfunipost
   \def\insertbslashes ##1##2##3##4##5{%
            \Bslash0##1##2\Bslash##3##4##5}%
   \def\gobbletwowords ##1 ##2 {}%
   \tmpnum=0
   \loop
      \sfcode\tmpnum=0 \advance\tmpnum by1
      \ifnum \tmpnum<128 \repeat
   \tmpnum=0
   \expandafter \pdfunidefA \pdfunichars {}%
}
```

Makro `\pdfunidef` nejprve zahájí skupinu a zapamatuje si jméno ⟨*makra*⟩, které má definovat, do `\tmpa`. Dále spustí `\pdfunipre` a expanduje parametr ⟨*text*⟩. Do pracovního makra `\out` bude postupně střádat výsledek konverze. Výchozí hodnota je `\pdfinitcode`, což je `\376\377`, neboli prefix UNICODE stringu. Dále je definováno pomocné makro `\odef`⟨*makro*⟩⟨*6 oktalových cifer*⟩⟨*mezera*⟩ poněkud trikoidním způsobem. Makro provede `\def`⟨*makro*⟩`{&`⟨*činnost*⟩`}`. Proč je tam znak `&`, vysvětlíme později. Makro přidá odpovídající oktalové cifry do výstupního makra `\out`, přitom k těmto cifrám přidá backslashe. Seznam `\pdfunipost` obsahuje jednotlivé příkazy `\odef`, tedy připraví makra ke konverzi.

Dále je připraveno pomocné makro `\insertbslashes`⟨*5 oktalových cifer*⟩, které přidá šestou nulu na začátek a vloží na správné místo backslashe. Makro `\gobbletwowords` odstraní dvě následující slova ukončená mezerou. Budeme to potřebovat za chvíli.

V závěru této části cyklus uloží každému znaku s kódem menším než 128 sfkód rovný nule. Podle této hodnoty sfkódu poznáme, že znak je z dolní ASCII tabulky a bude kódován na `\000`⟨*znak*⟩. Ostatní znaky vyjmenované v seznamu `\pdfunichars` budou mít každý svůj sfkód postupně se zvětšující od jedničky. To zajistí makro `\pdfunidefA`, které postupně ze seznamu `\pdfunichars` odlupuje jednotlivé znaky a dává jim odpovídající sfkódy.

```
\def\pdfunidefA #1{\if\relax#1\relax%
      \def\next{\let\next= }\afterassignment\pdfunidefB
         \expandafter\next\tmp\end
   \else \advance\tmpnum by1 \sfcode‘#1=\tmpnum \expandafter
        \pdfunidefA
   \fi
}
```

Postupné odlupování končí, když je načten prázdný parametr (poznáme to podle `\if\relax#1\relax`, to se totiž sejde první `\relax` s druhým). V takovém případě zahájíme postupné odlupování expandovaného ⟨*text*⟩, který máme v `\tmp`. Konec druhého řádku ukázky po provedení `\expandafter` a následné expanzi `\next` vypadá takto:

`\let\next=` ⟨*expandovaný text*⟩`\end`

Mezera za rovnítkem je důležitá, protože další případná mezera se skutečně přiřadí. Příkaz `\let\next` odloupne z ⟨*expandovaného textu*⟩ první token a po přiřazení spustí `\pdfunidefB`.

```
\def\pdfunidefB {%
   \ifcat x\noexpand\next \pdfunidefC
   \else \ifcat.\noexpand\next \pdfunidefC
   \else \ifcat\noexpand\next\space \edef\out{\out\pdfspacecode}%
   \else \ifcat^\noexpand\next \edef\out{\out\pdfcircumflexcode}%
   \else \ifcat_\noexpand\next \edef\out{\out\pdflowlinecode}%
   \else \expandafter\ifx\expandafter&\next
   \fi\fi\fi\fi\fi\fi
   \ifx\next\end \edef\next{\def\tmpa{\out}}\expandafter\egroup\next
      \else
      \def\next{\let\next= }\afterassignment\pdfunidefB
          \expandafter\next
   \fi
}
```

Makro `\pdfunidefB` prošetří obsah `\next` a podle jeho typu provede jeden krok konverze. V závěru své činnosti zavolá rekurzivně samo sebe a odloupne z ⟨*expandovaného textu*⟩ další token. To dělá tak dlouho, dokud nenarazí na `\end`. Až se tak stane, vloží do `\next`:

`\def`⟨*makro*⟩`{`⟨*zkonvertovaný výstup*⟩`}`

a nejprve expanduje `\next` pomocí `\expandafter` (tehdy si ještě pamatuje obsah `\next`) a vzápětí poté ukončí skupinu, vše tím zapomene a provede výsledek připravené expanze. Dílo je dokonáno.

Vraťme se k makru `\pdfunidefB`. Toto makro vyhodnotí především kategorii odloupnutého tokenu. Je-li to mezera, znak `^` nebo `_`, přidá do `\out` jejich odpovídající kód. Je-li to písmeno nebo jakýkoliv jiný znak, provede `\pdfunidefC`. Podívejme se tedy, co dělá `\pdfunidefC`:

```
\def\pdfunidefC {\edef\tmp{\expandafter\gobbletwowords\meaning\next}%
   \edef\out{\out\pdfunidefD}%
}
```

Makro `\pdfunidefC` potřebuje zpětně zjistit, jaký znak je v `\next` uložen. K tomu použije příkaz `\meaning\next`, který vypíše třeba `the character B`. Pomocí `\gobbletwowords` odstraníme dvě slova `the character` a v makru `\tmp` zbude jen `B`. Do `\out` pak přidáme výsledek expanze `\pdfunidefD`.

```
\def\pdfunidefD{%
   \ifnum\sfcode\expandafter`\tmp=0 \Bslash 000\tmp
   \else
      \expandafter \insertbslashes \ifcase
             \sfcode\expandafter`\tmp\pdfunicodes
         \else 000077\fi
   \fi
}
```

Toto makro musí pracovat jen s expandovatelnými primitivy. Nejprve zkontroluje, zda konvertovaný znak (skrytý v `\tmp`) má sfkód roven nule. V takovém případě expanduje na `\000`⟨*znak*⟩. Jinak pomocí `\expandafter` rozvine `\ifcase\sfcode`⟨*znak*⟩. Protože tato podmínka není terminovaná, expanduje se i tělo tohoto `\ifcase` skryté v makru `\pdfunicodes`. Víme, že tam je psáno

`\or`⟨*5ciferný kód*⟩`\or`⟨*5ciferný kód*⟩`\or`⟨*5ciferný kód*⟩...

Celý `\ifcase` je uzavřen pomocí `\else 000077\fi`, což je kód otazníku. Expandafter před `\ifcase` způsobí, že toto `\ifcase` expanduje na svou odpovídající větev podle sfkódu, kde je ⟨*5ciferný kód*⟩. Před tímto výsledkem se ovšem rozprostírá makro `\insertbslashes`, které do tohoto ⟨*5ciferného kódu*⟩ doplní první nulu a backslashe.

Zajímavý trik je na řádku těsně nad řadou `\fi\fi\fi\fi\fi\fi` v těle makra `\pdfunidefB`. Je tam řečeno `\expandafter\ifx\expandafter&\next`. Představme si, že `\next` je makro definované pomocí `\odef`. V takovém případě po vykonání těch dvou `\expandafter` dostáváme:

`\ifx&&`⟨*činnost*⟩

a provede se požadovaná činnost. Je-li `\next` cokoli jiného, není to shodné se znakem `&` a neprovede se nic. Je-li `\next` přímo znak `&`, neprovede se také nic.

Soubor `pdfuni.tex` je ukončen definicí makra z OPmac, která mění konvertor spuštěný při činnosti `\outlines`⟨*num*⟩ z původní hodnoty (konverze do ASCII) na novou hodnotu (konverze pomocí `\pdfunidef`).

```
\def\cnvhook #1#2{#2\pdfunidef\tmp\tmp} % the default convertor
                                        % in OPmac is redefined here
```

5. Literatura

[1] `http://www.adobe.com/devnet/acrobat/pdfs/pdf_reference_1-7.pdf`

[2] `http://petr.olsak.net/opmac.html`

Summary: PDFuni – accents in PDF bookmarks

Bookmarks in the PDF document (pdfoutlines) are texts which are displayed by PDF viewer under certain circumstances. They are clickable and connected to their destination in the document. They are organized in a tree structure. The bookmarks are never printed. Many TEX users may have noticed that Czech and Slovak texts are sometimes destroyed in the bookmarks: the accented letters are displayed badly. This article explains the technical background of this problem and describes the PDFuni macro package for plain TEX which solves this problem for Czech and Slovak texts in bookmarks and in the others PDF strings.

Key words

PDF, outlines, encoding, TEX, macro package, PDFuni