Zpravodaj Československého sdružení uživatelů TeXu

Petr Olšák

Makro na konverzi textů, PDF záložky

Zpravodaj Československého sdružení uživatelů TeXu, Vol. 16 (2006), No. 1, 26–32

Persistent URL: `http://dml.cz/dmlcz/150000`

Terms of use:

© Československé sdružení uživatelů TeXu, 2006

Institute of Mathematics of the Czech Academy of Sciences provides access to digitized documents strictly for personal use. Each copy of any part of this document must contain these *Terms of use*.

This document has been digitized, optimized for electronic delivery and stamped with digital signature within the project *DML-CZ: The Czech Digital Mathematics Library* `http://dml.cz`

Makro na konverzi textů, PDF záložky

Petr Olšák

Občas se mi při programování maker v TEXu stane, že potřebuji překonvertovat (zhruba řečeno) „řetězec znaků na jiný řetězec znaků“. Například potřebuji z textu sundat háčky a čárky, potřebuji text převést do jiného kódování, potřebuji si v textu všímat některých speciálních znaků a ty převést na jiné znaky... Už dlouhou dobu jsem si říkal, že by nebylo marné udělat na to nějaké univerzální makro. Posledním podnětem k tvorbě tohoto makra byl požadavek pana Kubena vyřešit problém s českými texty v záložkách v PDF při použití `hyperref` a bez použití `inputenc`.

V tomto článku představím makro `cnv.tex`, které umožňuje konverzi podle tabulky, kterou může deklarovat uživatel. Konverze se vyhýbá běžné expanzi a může jí podléhat jakýkoli token jakékoli kategorie.

Motivace

Pan Kuben mě požádal, ať se pokusím vyřešit problém s nefungující češtinou v PDF záložkách při použití balíčku `hyperref` (s volbou `unicode`) v LATEXu. Vysvětlil mi, že pokud navíc použije balíček `inputenc` (tj. akcentované znaky má aktivní), pak vše funguje, jak má. Pokud ale nechce aktivizovat česká písmena s háčky a čárkami a použije jen `\usepackage{czech}` bez balíčku `inputenc`, pak texty v záložkách nefungují. Přitom v CSLATEXu bychom se rádi balíčku `inputenc` vyhnuli, protože pak máme například méně starostí při zpracování rejstříků a máme čitelnější log soubory.

Po kratším průzkumu jsem přišel věci na kloub: makro `hyperref` používá pro texty do záložek 16bitový UNICODE a snaží se text do tohoto kódování poněkud amatérským způsobem konvertovat: před každý „obyčejný znak“ přidá nulový bajt a aktivní znaky nechá expandovat na požadované dva bajty podle deklarace v souboru `puenc.def`. Je tedy nabíledni, že pokud akcentované znaky nemáme aktivní, nebude to fungovat. Protože text pro záložku je už uložen v parametru (například při činnosti LATEXového makra `\section`), kategorie jednotlivým znakům jsou už přiděleny a znaky nelze jednoduše dodatečně „aktivizovat“. Chtělo by to tedy důkladnější makro na konverzi, které nespoléhá na žádnou expanzi ani na aktivní kategorie znaků. Tak jsem se konečně pustil do psaní dlouho odkládaného makra `cnv.tex`. Na „zakázce“ pro pana Kubena jsem pak makro odladil. Nakonec jsem makro zveřejnil k volnému použití [1].

Použití makra

Použití makra `cnv.tex` se pokusím ukázat právě na konverzi českého textu do PDF záložek. V předchozím odstavci jsem nebyl úplně přesný, když jsem psal, že text v záložkách je v 16bitovém UNICODE. Přesněji tento text obsahuje vesměs oktalovou notaci jednotlivých bajtů. V nezměněném tvaru zůstávají jen běžné ASCII znaky. Takže text „Záložka“ se konvertuje na:

```
\376\377\000Z\000\341\000l\000o\001\176\000k\000a
```

Prefix `\376\377` říká PDF prohlížeči, že text záložky je v UNICODE.

Chceme-li takovou konverzi použít, pak v csplainu můžeme psát:

```
\input cnv-pu
\newcount\numlink
\def\zalozka#1{\global\advance\numlink by1
   \pdfdest num\numlink fith\relax % deklarace cíle záložky
   \def\cnvtable{pu}\cnvin{#1}%    % konverze textu pro záložku
   \pdfoutline goto num\numlink{\string\376\string\377\cnvout}}
\zalozka{Tento text bude v záložce}
```

Ukázka makra je poněkud zjednodušená: například cíl záložky se většinou hodí „vysunout“ poněkud výše, aby to pěkně vypadalo. Nicméně způsob použití makra `cnv.tex` je zde zřejmý: nejprve načteme pomocí `\input` konverzní tabulku `pu` (PdfUnicode). Pokud ještě nebylo načteno samotné makro `cnv.tex`, pak se v tomto okamžiku načte automaticky také. Makro umí pracovat s libovolným množstvím konverzních tabulek. Je tedy potřeba před vlastní konverzí deklarovat, podle jaké tabulky konverze proběhne (`\def\cnvtable{pu}`). Samozřejmě, pokud makro používáme jen k jedinému účelu a s jedinou tabulkou, stačí napsat `\def\cnvtable{pu}` někam mezi globální deklarace. Výsledek konverze máme po provedení makra `\cnvin` připraven v makru `\cnvout`. Pomocí primitivu pdfTEXu `\pdfoutline` tento konvertovaný text umístíme do záložky včetně prefixu `\376\377`.

Možnosti deklarace konverzních tabulek jsou patrné po pohledu do souboru `cnv-pu.tex`. Konverze je dvouprůchodová: v prvním průchodu probíhá expanze těch tokenů, které jsou deklarovány jako makra příkazem `\predef`. Například úsek tabulky:

```
\predef \TeX    {TeX}
\predef \uv #1{\clqq #1\crqq}
```

způsobí, že token `\TeX` se v prvním průchodu expanduje na tři tokeny `TeX` a `\uv{text}` se expanduje na `\clqq␣text\crqq`. Pointa je ale v tom, že tyto deklarace vůbec nekolidují s běžným významem deklarovaných tokenů (např. sekvence `\TeX` je v TEXu definována poněkud jinak) a navíc můžeme takto deklarovat jakýkoli token jakékoli kategorie. Takže je možné:

```
\predef X{aha}
```

což způsobí, že každý výskyt znaku X se promění na aha. K tomuto výsledku „expanze“ se TEX při prvním průchodu vrací.

V druhém průchodu probíhá finální konverze, která se deklaruje pomocí `\findef` a má stejná pravidla, jako `\predef`. Rozdíl je v tom, že `\findef` nemůže pracovat s parametry a dále k výsledku expanze se už konverzní proces nevrací a považuje ho za definitivní. Takže `\predef␣X{(X)}` způsobí chybu (nekonečná rekurze), zatímco `\findef␣X{(X)}` je v pořádku a způsobí, že každý výskyt znaku `X` bude obalen závorkami.

V souboru `cnv-pu.tex` je tedy psáno:

```
\findef a{\string\000a}
\findef b{\string\000b}
\findef c{\string\000c}
... atd.
\findef \clqq {\string\000\string\214}
\findef \crqq {\string\000\string\215}
```

K dispozici je ještě další deklarační makro `\cnvaccent`. V `cnv-pu.tex` například můžeme najít:

```
\cnvaccent  \'A{\string\000\string\301}
```

Tento zápis způsobí, že každý výskyt `\'A` nebo `\'␣A` nebo `\'{A}` se v druhém průchodu konverze promění na `\000\301`. Navíc se makro `\cnvaccent` už v době deklarace pokusí expandovat `\'{A}` a pokud je výsledkem jediný token (například Á), pak výše uvedená deklarace provede interně ještě:

```
\findef Á{\string\000\string\301}
```

kde Á je výsledek expanze `\'{A}`. Je tedy vhodné před načtením příkazu `\cnvaccent` zařídit správnou expanzi maker `\'`, `\v` atd. To je v tabulce `cnv-pu.tex` provedeno takto:

```
\ifx\fontencoding\undefined
   \csname csaccents\endcsname
\else
   \fontencoding{\encodingdefault}\selectfont
\fi
```

Je zde pamatováno nejen na csplain (provede se příkaz `\csaccents`), ale i na LATEX (provedou se příslušné příkazy z NFSS). Tabulka je čtena celá uvnitř skupiny, takže tyto změny neohrozí případné jiné nastavení uživatele. Přitom vlastní deklarace jsou globální, takže po ukončení skupiny se neztratí.

Proč je to tak uděláno? Tabulka `cnv-pu.tex` konvertuje z interního 8bitového kódování TEXu na oktalový zápis podle UNICODE, ale toto interní kódování

může být jakékoli a není o něm v tabulce explicitní zmínka. Nemusíme tedy mít desítky tabulek typu `cork-pu`, `iso2-pu` atd.

Nebudu zde podrobně rozepisovat další vlastnosti makra `cnv.tex` a možnosti deklarací konverzních tabulek. Vše je dokumentováno přímo v souboru `cnv.tex`. Stručně jen uvedu některé vlastnosti:

- Můžeme deklarovat implicitní konverzi pro nedeklarované tokeny (viz `\cnvdefault`, `\cnvadefault`).
- Je možno nastavit, zda deklarační makra pracují jako `\def`, `\edef`, `\gdef` nebo `\xdef` (viz `\predefmethod`, `\findefmethod`).
- Makro se vyrovná s konverzemi mezer i svorek jakoby se jednalo o obyčejné znaky (kdo psal nějaké složitější TEXové makro, ten ví, že to zcela obyčejné znaky nejsou).
- Lze během konverze spustit některá makra běžným způsobem a nechat je zpracovat třeba i hlavním procesorem TEXu (`\cnvexec`, `\cnvnext`, `\cnvexpand`, `\cnvexpandtext`).
- Je možné deklarovat a používat libovolné množství tabulek (`\cnvtable`).

Původní „zakázka“ souvisela s balíčkem `hyperref` v LATEXu. Nevytvořil jsem sice samotný balíček, který problém řeší, ale pokusil jsem se aspoň zveřejnit „radu“ pro LATEXové uživatele, co mají napsat na začátek svého dokumentu:

```
\usepackage{czech}
\input cnv-pu
\usepackage{hyperref}  % volba [unicode] není nutná
\def\pdfstringdef #1#2{% \pdfstringdef předefinujeme
    \bgroup \escapechar=‘\\%
    \def\cnvtable{pu}\cnvin{#2}%
    \xdef #1{\string\376\string\377\cnvout}\egroup }
```

Vidíme, že je zde předefinováno interní makro `\pdfstringdef` použitého balíčku `hyperref`. Proto je samozřejmě nutné příkaz `\def\pdfstringdef` umístit až po volání balíčku `hyperref`. Dále je potřeba provést `\input cnv-pu` až po `\usepackage{czech}`, aby bylo správně nastaveno makro `\encodingdefault` a deklarace `\cnvaccent` proběhly podle zvoleného kódování uživatele.

Pan Kuben se pokusil navázat spojení s autorem balíčku `hyperref` a požádal ho, aby tam využití makra `cnv.tex` zapracoval například jako další volbu balíčku `hyperref`. Zdá se, že tato iniciativa vyšla naprázdno.

Rozčarování

Tušil jsem, že převod záložek do UNICODE výše popsaným způsobem nebude všelék na všechny neduhy a problémy s češtinou v záložkách. Tušení mě, bohužel,

nezklamalo. Existence neportabilních záložek v Portable Document Formátu zůstává a zůstane nadále, protože v tomto případě vše závisí na prohlížeči a hlavně na použitém systémovém fontu pro záložky, který nemůžeme jako autoři dokumentu ovlivnit.

Panu Kubenovi to zřejmě na jeho instalaci fungovalo (v Acroreaderu 6 a 7 ve W2000 a WinXP). Mně ale Acroreader zobrazil češtinu v záložkách i při UTF8 kódování chybně. V Linuxovém Acroreaderu 5 (ve Slackware 8 i 9) se mi místo všech akcentovaných znaků objevily jen puntíky. V takovém případě je čitelnější text, při kterém nechám v záložkách kódování ISO-8859-2, protože pak se mi zmrví jen znaky, které se v ISO-2 liší od ISO-1. Zkusil jsem také Linuxový Acroreader 7. Tam byly české znaky v UNICODE záložkách zobrazeny správně, ale ne všechny. Místo „Č“ jsem viděl takový podivný čtvereček a místo uvozovek taky. Zato „ž“ se zobrazovalo správně. Prohlížeč má možnost měnit velikost písma v záložkách. Při této změně začnou (podle očekávání) zlobit zase jiná písmena ve stejných textech. Budeme se tedy asi muset smířit s tím, že zobrazování češtiny v záložkách při použití Acroreaderu je nekonečný problém. Myslet si, že když já to vidím ve svém PDF prohlížeči dobře, že to tak uvidí celý svět, je poněkud naivní. Pan Kuben generuje PDF soubory s českými texty v záložkách asi pro svou soukromou potřebu nebo s vědomím, že to není dobře čitelné všude. Ani jemu to nefunguje všude, například píše, že v Acroreaderu 4 a 5 v OS/2 a eCS je to hodně špatné.

Na základě těchto zkušeností jsem se rozhodl, že zatím nebudu nově generovat například dokument `tbn.pdf` tak, aby měl texty záložek v UNICODE. Doba k tomu ještě zdaleka nenazrála. Nechám své PDF dokumenty tak, jak jsou (v ISO-8859-2), s vědomím, že ne všichni vidí texty záložek dobře.

Další možnosti

Koverze textů PDF záložek je jen jedna z možností využití makra `cnv.tex`. Tušíme, že díky nastavovatelným konverzním tabulkám můžeme makro použít pro mnoho jiných účelů. Bohužel ale makro pracuje jen s jednotlivými znaky, které v prvním průchodu mohou expandovat a v druhém průchodu se mohou proměnit na konečný výsledek. Napadlo mě vytvořit nadstavbu nad tímto makrem, které si všímá nejen jednotlivých znaků, ale celých úseků znaků (slov) a tyto úseky mění podle požadavku uživatele na případně jiná slova. Tak vzniklo makro `cnv-word.tex`.

Konverzi slov zajišťuje deklarační makro `\stringdef`, které se používá následovně:

```
\stringdef {slovo}   {nono}
\stringdef {slož}    {odhod}
\stringdef {sl}      {SL}
```

```
\wconvert {Vysloužilý voják složil zbraně a neřekl ani slovo.}
```

V makru `\cnvout` pak máme: „VySLoužilý voják odhodil zbraně a neřekl ani nono.“.

Na pořadí deklarací záleží. Kdybychom použili `\stringdef␣{sl}␣{SL}` jako první, pak by se na další slova začínající na `sl` už nedostalo a výsledek by byl následující: „VySLoužilý voják SLožil zbraně a neřekl ani SLovo.“.

Jak je toto makro uděláno je podrobně popsáno v souboru `cnv-word.tex`. Rád uvítám další náměty na vylepšení. Bohužel, možnosti regulárních výrazů pomocí makrojazyka TEXu asi nenaprogramujeme. Na určité věci je asi potřeba použít jiný nástroj.

Odkaz

[1] `ftp://math.feld.cvut.cz/pub/olsak/makra/*`:
`cnv.tex` – vlastní makro včetně dokumentace,
`cnv-pu.tex` – konverzní tabulka pro PDF záložky,
`cnv-word.tex` – experiment s konverzí celých slov.

Summary: Macro for conversion of texts, PDF outlines

A TEX macro programmer sometimes needs to convert a string of tokens to another string of tokens with defined rules. For example, we need to remove accents from a Czech text or to recode this text to another encoding or to transform some special characters to something else or... For this purpose, the `cnv.tex` macro was designed. Last but not least motivation was the problem of my TEX colleague Jaromír Kuben: he needed to convert Czech text to PDF outlines (conversion to UNICODE) using `hyperref.sty` but without activating Czech characters by `inputenc.sty`.

This article presents the `cnv.tex` macro for string conversion by a user defined table. The conversion process is out of normal expansion and each token (independent on category code) can be converted to a single token or a group of tokens. The macro is available including documentation on `ftp://math.feld.cvut.cz/pub/olsak/macros/` in files `cnv.tex`, `cnv-pu.tex`, and `cnv-word.tex`.

Redakční poznámka

Regulární výrazy jsou velmi mocným nástrojem a uživatelům TEXu (nebo alespoň tvůrcům makrobalíků) často scházejí k efektivní práci. Proto současný vývojářský tým pdfTEXu zakomponoval od verze 1.30 POSIX regex knihovnu. Zatím

jde o experimentální kód, jehož uživatelské rozhraní se ještě může měnit. Cituji za základního použití:

```
\pdfmatch [icase] [subcount <number>}] {<pattern>}{<string>}
     It returns the same values as \pdfstrcmp, but with the
     following semantics:
       -1: error case (invalid pattern, ...)
        0: no match
        1: match found
     Options:
     * icase: case insensitive matching
     * subcount: it sets the table size for found subpatterns.
       A number -1 resets the table size to the start default.
\pdflastmatch <number>
     The result of \pdfmatch is stored in an array.
     The entry 0  contains the match, the following entries
     submatches. The positions of the matches are also available.
     They are encoded:
       <position> -> <match string>
     The position -1 with an empty string indicates
     that this entry is not set.
```

Zároveň se testuje verze 1.40, do níž je integrován jednoduchý, ale plnohodnotný skriptovací jazyk lua (http://www.lua.org). Rozhraní může číst a zapisovat do registrů TEXu. TEXtak získá regulární výrazy, aritmetiku v plovoucí desetinné čárce, snazší programování cyklů ap.

Vít Zýka