Zpravodaj Československého sdružení uživatelů TeXu

Oldřich Ulrych

Z databáze do TeXu aneb dozvuky EuroTeXu 92

Zpravodaj Československého sdružení uživatelů TeXu, Vol. 2 (1992), No. 4, 164–176

Persistent URL: http://dml.cz/dmlcz/149644

Na závěr mi dovolte popřát vám všem (v této roli naposled) vše nejlepší v tomto novém roce, zejména hodně zdraví, pevných nervů, úspěchů v práci i osobním životě, zkrátka vše ff primisima OK, a v neposlední řadě i pěkné chvíle s TEXem.

-jv

Z databáze do TEXu aneb dozvuky EUROTEXu 92

Oldřich Ulrych

Tento příspěvek vznikl jako (prudká) reakce na dění během přípravy konference EUROTEX 92 a jeho cílem je popsat postup, který se nám jeví jako optimální pro přípravu různých TEXových výstupů z databáze. Během přípravy na konferenci byla udržována databáze účastníků, přičemž z této databáze bylo nutno tisknout různé seznamy (většinou v abecedním pořadí účastníků). Průběžně se měnil nejen obsah databáze (což je přirozené), ale také její struktura neboť v průběhu přípravy se ukázalo, že je potřeba mít o účastnících více informací, než jsme předpokládali na začátku. Problém, který chceme tímto příspěvkem vyřešit, je minimalizace práce při neustále se měnících podmínkách.

Idea celého postupu je tato: Napišme jednou pro vždy obecný program pro databázový systém, se kterým pracujeme (požadavky na databázový systém budou uvedeny níže). Tento program budeme nazývat `dbf2tex.prg` a jeho verzi pro FoxBase+ (nebo dBase III+) uvedeme na konci. Jestliže budeme chtít obsah libovolné databáze vytisknout TEXem, spustíme na tuto databázi program `dbf2tex.prg`. Tím vznikne pomocné makro pro TEX a pomocný program pro databázi. Spuštěním tohoto pomocného programu na databázi získáme jakožto zvláštní textový soubor úplný obsah databáze, přičemž obsah každého pole je argumentem řídicího slova odvozeného z názvu pole. Každý záznam databáze je obklopen řídicími slovy začátku a konce záznamu. Pro konkrétní výstup již stačí napsat pouze příslušné makro pro TEX. Výběr položek, které se tisknou,

je až na úrovni zpracování ASCII souboru TEXem (při stejném uspořádání záznamů je tedy možné snadno dosáhnout různých typů výstupů).

Výhody: Minimalizace prací na úrovni databáze; pro jakýkoliv výstup nemusíme psát žádný databázový program;

z téhož ASCII souboru můžeme snadno vysázet různé seznamy (s výběrem různých informací);

udržujeme minimum souborů (pouze obecný databázový program — který m.j. obsahuje obecnou část makra pro TEX — a pro každou databázi jediné TEXové makro), pomocí nichž se dá kdykoliv jakýkoliv výstup znovu vytvořit;

konkrétní výstup (pokud není potřeba třídit databázi podle různých kritérií) se volí jediným řídicím slovem na úrovni TEXu.

Nevýhody: Nutno používat takové databázové systémy, které umožňují napsání obecného programu (viz níž);

do textového souboru se z databáze zapisuje celá databáze, což může být časově náročnější především na pomalých počítačích nebo při rozsáhlých databázích;

TEXem se pokaždé zpracovává větší soubor, než by tomu bylo při napsání speciálních databázových programů pro každý jednotlivý výstup zvlášť (tato nevýhoda je relativní, neboť vždy lze pomocný databázový program editací jednoduše upravit tak, aby výstupní soubor obsahoval pouze nutná pole databáze);

názvy polí databáze se mohou skládat pouze z písmen;

jednou zavedené názvy polí databáze by již neměly být měněny, abychom nemuseli přepisovat příslušné názvy v TEXovém makru.

Volba databázového systému

Aby bylo možné výše popsaný postup téměř automaticky provádět, je nutné, aby databázový systém, se kterým pracujeme, měl určité vlastnosti. Základními dvěma kritérii pro volbu databáze jsou programovatelnost a splnění našich požadavků, kvůli kterým chceme udržovat data v databázi. Další požadavky jsou:

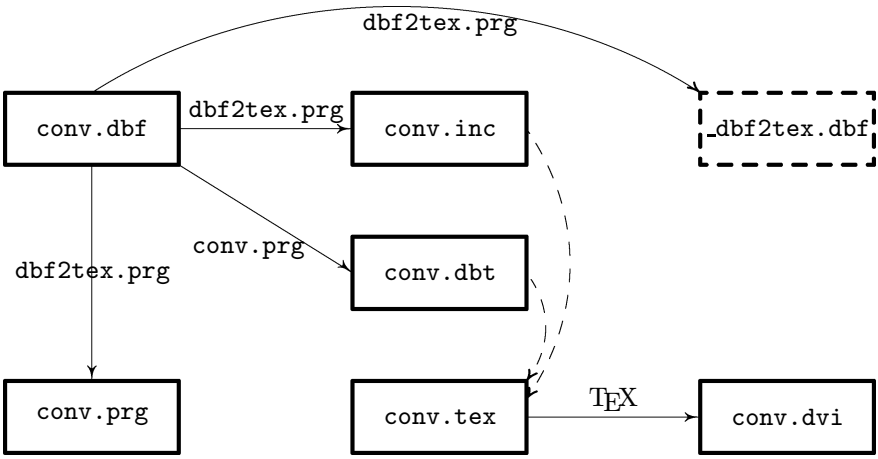
1. Možnost zjistit název zpracovávané databáze (název vlastního souboru bez extenze).
2. Možnost zjistit strukturu databáze (počet polí, jejich typ a názvy).
3. Výstup textu do souboru (a to včetně některých speciálních znaků — jestliže například znak " je vyhrazen v databázi pro vyznačování

řetězců znaků, musí existovat možnost, jak tento znak zapsat do výstupního souboru).

Schéma celkového postupu

Celkový postup doporučujeme provádět podle schématu, které popíšeme níže pro databázový soubor, jehož název jsme si zvolili `conv.dbf`. Schématicky je celý postup znázorněn na obrázku níž. Předpokládáme, že celý postup z databáze do TEXu bude probíhat na jednom počítači, tj. na počítači, kde je instalován jak TEX, tak i databáze, se kterou pracujeme. Protože přiložený program `dbf2tex.prg` je psán v programovacím jazyce pro FoxBase+, je nutno jej upravit podle databázového systému, který používáte (ve FoxBase+ existuje pět typů polí, a to řetězce, číselné a logické proměnné, dále proměnné typu datum a poznámka). V programu `dbf2tex.prg` je možné také nastavit cesty pro ukládání textových souborů pro TEX do adresářů, do kterých jsou nastaveny cesty pro TEX.

Obrázek ukazuje postup, jakým jsou jednotlivé soubory vytvářeny. Přerušované čáry naznačují, že soubory, ze kterých vycházejí, jsou začleněny příkazem `\input` do jiného souboru.



1. Při vytváření či modifikacích struktury databáze `conv.dbf` je nutné dodržet pravidlo, že názvy polí databáze se skládají pouze z pís-

men. Pokud je databáze `conv.dbf` již vytvořena, je nutno ověřit, že všechny názvy polí se skládají pouze z písmen. Pokud tomu tak není, nahradíme názvy obsahující jiné znaky než písmena názvy sestavenými pouze z písmen (tyto názvy budou používány jako řídicí slova v TEXu). Uvědomíme si, že pokud existuje program `conv.prg`, pak tento program bude přepsán výstupem z programu `dbf2tex.prg`. Programem `dbf2tex.prg` vytvoříme pomocné makro `conv.inc` pro TEX a pomocný program `conv.prg`. Během práce programu `dbf2tex.prg` se vytváří pomocná databáze s názvem `_dbf2tex.dbf`, která se před ukončení programu `dbf2tex.prg` smaže.

2. Vždy, když potřebujeme aktuální výpis získaný z databáze pomocí TEXu, doplníme a přerovnáme databázi podle potřeby (zachováme jméno `conv.dbf`) a spustíme program `conv.prg`. Tím vznikne soubor `conv.dbt` v adresáři přístupném pro TEX. Pravidla pro vytváření tohoto souboru jsou popsána níž.
3. Vytvoříme nebo doplníme soubor `conv.tex` o definice nového formátu výstupu. Postup při vytváření tohoto souboru je popsán níže.
4. Doplníme řídicí slova definující nový formát v předchozím bodě v souboru `conv.tex` někam před závěrečné řádky
   ```
   \input conv.dbt
   \bye.
   ```
   Tím vzniká nová volba pro zpracování celé databáze. Vybereme pouze jednu volbu, ostatní označíme jako komentář a soubor `conv.tex` zpracujeme TEXem. Příklad takového souboru bude uveden níž.

Co je potřeba znát o `conv.dbt`

Každý záznam je uvozen řídicím slovem `\recordbegin` a ukončen řídicím slovem `\recordend`. Obsah logických polí je vypsán do výstupního souboru pouze v případě, že hodnota logického pole je 1 (true) a to tak, že do výstupního souboru je zapsáno řídicí slovo sestávající z názvu pole s připojeným slovem `true`. Všechny ostatní obsahy polí jsou ve výstupním souboru uzavřeny ve složených závorkách za řídicími slovy, která jsou odvozena z názvu pole. U položek typu řetězec jsou vypisovány pouze neprázdné položky, přičemž jsou odstraněny případné mezery na začátku a na konci pole. Datum je vypisováno ve tvaru `dd.mm.yy` (den, měsíc, rok — odděleno tečkami). Znalost této struktury je nutná pro psaní výstupního makra.

Jak postupovat při vytváření souboru `conv.tex`.

Doporučujeme postup, při kterém zavedeme nové řídicí slovo pro každý typ výstupu. Toto umožňuje zvolit požadovaný tvar výstupu volbou pouze jednoho řídicího slova.

V rámci tohoto řídicího slova uvedeme jednak základní parametry výstupu (rozměry stránky, typy písma, ...) a jednak řídicí slovo `\useddefs`, za kterým musí být ve složených závorkách uveden seznam názvů polí databáze (s výjimkou polí logického typu), která budou zpracovávána. Pokud uvádíme opravdu jen ta pole, která budeme používat, je tím mírně zrychleno zpracování. Dále je tím řečeno, že na začátku zpracování jsou řídicí slova sestávající z uvedených názvů řídicích slov známa (a ekvivalentní řídicímu slovu `\relax`). Všechny logické proměnné mají na začátku souboru hodnotu `false` (bez ohledu na to, zda budou či nebudou použity).

Dále by měla být v rámci nového řídicího slova předefinována řídicí slova `\recordbegin` a `\recordend`. Měla by jednak obsahovat vyznačení skupiny (`\begingroup` a `\endgroup`), v jejímž rámci se vysázejí požadované informace do požadovaného formátu. Tam, kde potřebujeme uvést obsah některé položky, použijeme místo ní řídicí slovo tvořené jejím názvem (toto platí s výjimkou logických polí databáze). O hodnotě logických polí lze rozhodovat pomocí podmínky `\if`⟨*pole*⟩, kde ⟨*pole*⟩ je název příslušného pole. Tento postup bude patrný z příkladu níž. Seznam názvů všech polí je vždy možné nalézt na konci souboru `conv.inc`.

V makru `conv.inc` je také definováno řídicí slovo `\testempty`, které má čtyři argumenty. Jestliže šířka horizontálního boxu, vytvořeného z prvního argumentu, je větší, než je rozměr uvedený jako druhý argument, vloží se do textu třetí argument, jinak se vloží čtvrtý argument. Toto řídicí slovo je tedy možné používat v makru `conv.tex` a činnost tohoto řídicího slova bude také patrná z ukázek dále.

Příklad databáze a jejího zpracování

Předpokládejme, že jsme vytvořili nebo že máme velmi jednoduchou databázi `conv.dbf`, která má následující strukturu:

```
    field name  type      width  dec
    ================================
1   FIRSTNAME   Character   20     0
2   SURNAME     Character   20     0
3   CAR         Logical      1     0
```

```
  4  BIRTHDAY    Date          8    0
  5  PAID        Numeric       8    2
```

Dále předpokládejme, že tato databáze má tři záznamy. Tyto záznamy (některá pole jsou úmyslně nevyplněná a všimněme si, že rok je zadáván také pouze posledním dvojčíslím) obsahují následující údaje:

```
FIRSTNAME Isaac       FIRSTNAME                 FIRSTNAME Saavedra
SURNAME   Newton      SURNAME   van Beethoven   SURNAME   de Cervantes
BIRTHDAY  01/05/43    BIRTDAY   12/16/70        BIRTHDAY  09/09/47
WITHCAR               WITHCAR   T               WITHCAR   T
PAID        200.00    PAID         00.00        PAID        100.00
```

Dále předpokládejme, že z této databáze budeme chtít vytisknout dva seznamy. První seznam by měl být abecedně seřazený podle příjmení a měl by obsahovat pouze příjmení a jména:

van Beethoven . . . . . .
Newton . . . . . Isaac
de Cervantes . . . . . Saavedra

Druhý seznam by měl být abecedně seřazený podle jmen a měl by obsahovat všechny informace o všech lidech, u kterých je logická položka WITHCAR nastavená na logickou hodnotu `true` (z data narození je vysázen pouze měsíc a rok):

van Beethoven . . . . . 12.1970 . . . . . 0.00
de Cervantes Saavedra . . . . 9.1947 . . . . . 100.00

Nyní popíšeme posloupnost kroků, které vedly k získání těchto dvou seznamů.

1. Vytvořili jsme soubor `conv.tex`. V tomto makru jsme definovali dvě řídicí slova, kterými budeme určovat typ seznamu, jenž se má sázet. Toto makro bylo napsáno pro plainTEX pomocí prostředků plainTEXu (o psaní definic je možné se dočíst např. v [1], [2]) a pro lepší představu je zde uvádíme celé:

```
\input conf.inc

% seznam příjmení a jmen oddělených tečkami

\def\onlynames{%
      \useddefs{FIRSTNAME,SURNAME} % použitá pole z databáze
      \hsize=80mm                  % šířka výstupní stránky
      \let\recordbegin\onlynamesb  % předefinování začátku záznamu
```

```
      \let\recordend  \onlynamese  % předefinování konce záznamu
      }
\def\onlynamesb{\begingroup} % začátek záznamu -
                             % všechny změny budou lokální
\def\onlynamese{%            % konec záznamu
   \line{\SURNAME\hdotsfill\FIRSTNAME}% do řádku příjmení, tečky, jméno
      \endgroup}         % všechny změny uvnitř  záznamu jsou lokální
\def\hdotsfill{\leaders\hbox to1em{\hss.\hss}\hfill}

% seznam všech informací o lidech s pravdivou položkou WITHCAR

\def\completelist{\useddefs{FIRSTNAME,SURNAME,PAID,BIRTHDAY}
                                                % bude použito vše
      \let\recordbegin\clistb   % předefinován začátek záznamu
      \let\recordend  \cliste   % předefinován konec záznamu
      }
\def\clistb{\begingroup} % začátek záznamu - všechny změny budou lokální
\def\cliste{\ifWITHCAR   % konec záznamu, záznamy s pravdivým polem WITH-
CAR
   \line{%               % do řádku sázet:
      \testempty{\SURNAME}{0pt}{\SURNAME\ }{}% jestliže příjmení je
                           % neprázdné, vysázet je
                           % spolu s jednou mezerou
      \FIRSTNAME\hdotsfill  % pak vysázet jméno a tečky
      \hbox to 20truemm{\expandafter\onlymy\BIRTHDAY A\hfil}%
                                       % pouze měsíc a rok
      \hbox to 20truemm{\hdotsfill}%      % tečky
      \hbox to 20truemm{\PAID\hfil}%      % zaplacená částka
    }
  \fi
  \endgroup}                % konec skupiny na konci záznamu.
\def\onlymy#1.#2.#3A{\ifnum#2=0 \else #2.#3\fi}

%     možné volby úpravy výstupu, definované v souboru conf.mac

%\onlynames     % seznam příjmení a jmen oddělených tečkami
%\completelist  % seznam všech informací o lidech s WITHCAR

%     zpracovávaná databáze

\input conf.d2t

\bye
```

2. Ve FoxBase+ jsme spustili na databázi `conv.dbf` program `dbf2tex.prg`.
3. Ve FoxBase+ jsme přerovnali databázi `conv.dbf` podle příjmení (pole SURNAME).
4. Ve FoxBase+ jsme spustili na takto přerovnanou databázi `conv.dbf` program `conv.prg`.
5. V souboru `conv.tex` jsme odstranili komentář před řídicím slovem `\onlynames` a soubor `conv.tex` jsme zpracovali TEXem. Tím jsme získali první seznam.
6. Ve FoxBase+ jsme přerovnali databázi `conv.dbf` podle jmen (pole FIRSTNAME).
7. Ve FoxBase+ jsme spustili na takto přerovnanou databázi `conv.dbf` program `conv.prg`.
8. V souboru `conv.tex` jsme odstranili komentář před řídicím slovem `\completelist` (řídicí slovo `\onlynames` jsme označili jako komentář) a soubor `conv.tex` jsme zpracovali TEXem. Tím jsme získali druhý seznam.

Z výše uvedeného je patrné, jak bychom postupovali, kdybychom potřebovali z dané databáze získat nějaký jiný seznam (rozšířit soubor `conv.tex` o další definice), či seznam po aktualizaci obsahu databáze (spustit program `conv.prg` a opět zpracovávat soubor `conv.tex` s příslušnou volbou TEXem), či seznam po změně struktury databáze (spustit program `dbf2tex.prg` a pak program `conv.prg`).

Program `dbf2tex.prg` pro FoxBase+ v. 2.1

Nyní již zbývá jen uvést program `dbf2tex.prg`, který používáme pro výše popsaný účel ve FoxBase+ (verze 2.1) nebo dBase III+. Tento program obsahuje tu část makra `conv.inc`, která je pro všechny databáze stejná. Zde je celý výpis programu `dbf2tex.prg`:

```
set talk off

* Common setting

pathfortex=""           && path for output ASCII files
texmacroex=".inc"       && extension of file with file dependent defs
texfileext=".d2t"       && extension of file with contents of database
dbprgext  =".prg"       && extension of programs in database
auxfile   ="_dbf2tex"   && auxiliary database
```

```
auxfiledbf=auxfile+".dbf"

* Useful constants not necessary to change

strsname ="\newstrings"
numsname ="\newnumbers"
datesname="\newdates  "
logsname ="\newifs    "
memosname="\newmemos  "

* Separation of databaze file name

ap=chr(34)              && apostrophe
dbfx=dbf()
do while '\'$dbfx
  dbfx=substr(dbfx,at('\',dbfx)+1)
enddo
dbfname=dbfx
dbfx=left(dbfx,at('.',dbfx)-1)
dbfp=dbfx+dbprgext
if len(dbfx)=0 then
      ?" Some database must be in use before you call this program!"
      return
endif

* Creation of database dependent program

copy to &auxfile structure extended

use &auxfile
goto top

set alte to &dbfp
set alte on
?"set talk off"
?"set alte to "+pathfortex+dbfx+texfileext
?"set alte on"
?"goto top"
?"do while .not.eof()"
?"  ?"+ap+"\recordbegin"+ap
do while .not.eof()
   do case
     case field_type='C'
     ?"  if len(trim("+field_name+"))>0 then"
     ?"  ?"+ap+"\a"+field_name+"{"+ap+"+trim("+field_name+")+"+ap+"}"+ap
```

```
     ?"  endif"
     case field_type='L'
     ?"  if "+field_name+" then"
     ?"    ?"+ap+"\"+trim(field_name)+"true"+ap
     ?"  endif"
     case field_type='N'
     ?"  ?"+ap+"\a"+field_name+"{"+ap+","+field_name+","+ap+"}"+ap
     case field_type='D'
     ?"  ?"+ap+"\a"+field_name+"{"+ap+"+trim(str(day("+field_name
     ??"),2))+"ap+"."+ap+"+trim(str(month("+field_name+"),2))+"
     ??ap+"."+ap+"+trim(str(year("+field_name+"),4))+"+ap+"}"+ap
     case field_type='M'
     ?"  ?"+ap+"\a"+field_name+"{"+ap+","+field_name+","+ap+"}"+ap
   endcase
   skip 1
enddo
?"  skip 1"
?"  ?"+ap+"\recordend"+ap
?"  ?"+ap+"%---------------"+ap
?"enddo"
?"?"+ap+"\endinput"+ap
?"?"
?"set alte off"
?"set talk on"
set alte off

* Database dependent definition file

texfile=pathfortex+dbfx+texmacroex
set alte to &texfile
set alte on

* TeX macro at the begin of the file

?"% All modification of this file will be lost after next run "
?"% of DBFTOTEX program! Therefore any permanent changes of definitions"
?"% below must be done in DBFTOTEX.PRG file (but it is not recommended)."
?"% For local definitions use separate file"+dbfx+".tex"
?
?"\edef\catcodeat{\the\catcode`\@ } \catcode`\@=11"
?"\def\newif#1{\count@\escapechar \escapechar\m@ne"
?"  \expandafter\expandafter\expandafter"
?"   \edef\@if#1{true}{\let\noexpand#1=\noexpand\iftrue}%"
?"  \expandafter\expandafter\expandafter"
?"   \edef\@if#1{false}{\let\noexpand#1=\noexpand\iffalse}%"
```

```
?"  \@if#1{false}\escapechar\count@} % the condition starts out false"
?"\newbox\auxbox@"
?"\def\newifs    #1{\let\nexti\@newif\scanlist#1,,\relax}"
?"\def\@newif#1{\expandafter\newif\csname if#1\endcsname\scanlist}"
?"\def\newstrings#1{\let\nexti\@onedef\scanlist#1,,\relax}"
?"\def\@onedef#1{\expandafter\def\csname a#1\endcsname##1{}\scanlist}"
?"\let\newnumbers=\newstrings"
?"\let\newdates  =\newstrings"
?"\let\newmemos  =\newstrings"
?"\def\useddefs#1{\let\nexti\@useddef\scanlist#1,,\relax}"
?"\def\@useddef#1{\expandafter\def\csname a#1\endcsname##1%"
?"    {\expandafter\def\csname#1\endcsname{##1}}%"
?"    \expandafter\let\csname#1\endcsname\relax"
?"    \scanlist}"
?"\def\scanlist#1,{\testempty{#1}{0pt}{\let\next\nexti}%"
?"{\let\next\@attorelax}%"
?"    \next{#1}}"
?"\def\@attorelax#1\relax{}"
?"\def\testempty#1#2#3#4{\def\t@mporarya{#3}\def\t@mporaryb{#4}%"
?"    \setbox\auxbox@=\hbox{\ignorespaces#1\unskip}%"
?"    \ifdim\wd\auxbox@>#2{}\expandafter\t@mporarya\else"
?"                \expandafter\t@mporaryb\fi}"
?"\catcode`\@=\catcodeat              \let\catcodeat=\undefined"
?
?"\def\recordbegin#1\recordend{}"
?
?"% List of usable control sequences ("+dbfx+" database dependent)"
?
* Database file dependent contribution

goto 1
strslist=""
numslist=""
dateslist=""
logslist=""
memoslist=""
lorow=50
do while .not.eof()
   do case
     case field_type='C'
       strslist=strslist+','+trim(field_name)
       if len(strslist)>lorow
         ? strsname+'{'+substr(strslist,2)+'}'
         strslist=""
       endif
```

```
      case field_type='N'
        numslist=numslist+','+trim(field_name)
        if len(numslist)>lorow
          ? numsname+'{'+substr(numslist,2)+'}'
          numslist=""
        endif
      case field_type='D'
        dateslist=dateslist+','+trim(field_name)
        if len(dateslist)>lorow
          ? datesname+'{'+substr(dateslist,2)+'}'
          dateslist=""
        endif
      case field_type='L'
        logslist=logslist+','+trim(field_name)
        if len(logslist)>lorow
          ? logsname+'{'+substr(logslist,2)+'}'
          logslist=""
        endif
      case field_type='M'
        memoslist=memoslist+','+trim(field_name)
        if len(memoslist)>lorow
          ? memosname+'{'+substr(memoslist,2)+'}'
          memoslist=""
        endif
    endcase
    skip 1
enddo
if len(strslist)>0 then
  ? strsname+'{'+substr(strslist,2)+'}'
endif
if len(numslist)>0 then
  ? numsname+'{'+substr(numslist,2)+'}'
endif
if len(dateslist)>0 then
  ? datesname+'{'+substr(dateslist,2)+'}'
endif
if len(logslist)>0 then
  ? logsname+'{'+substr(logslist,2)+'}'
endif
if len(memoslist)>0 then
  ? memosname+'{'+substr(memoslist,2)+'}'
endif
?"\endinput"
?
set alte off
```

```
set talk on

use &dbfname
delete file &auxfiledbf
```

Literatura

[1] Knuth D. E.: *The TEXbook,* Amer. Math. Soc. & Addison Wesley Publ. Co., 1990.

[2] Doob M.: *Jemný úvod do TEXu,* překlad J. Daneš, J. Veselý, Univerzita Karlova, Praha 1990.

Oldřich Ulrych

Tvorba rejstříku

Zdeněk Wagner

Úvod

V tomto článku se budeme zabývat problematikou tvorby rejstříků pomocí programu ***MakeIndex***. Tento program byl původně vytvořen pro použití v LaTeXu, ale také jej lze využít v Plain TeXu. LaTeX má již nadefinovaná makra pro spolupráci s ***MakeIndex***em. Chceme-li vytvořit rejstřík v Plain TeXu, musíme si vše udělat sami. V každém případě je vhodné vědět, jak taková makra pracují, abychom mohli vytvářet libovolné speciality.

Součástí distribuce ***MakeIndex***u je podrobná dokumentace. Nebudu ji tudíž zde opisovat. Nejprve se tedy zmíníme o zvláštnostech tvorby rejstříku v češtině a slovenštině. Dále si vysvětlíme, jak ***MakeIndex*** spolupracuje s LaTeXem a jak lze vytvořit rejstřík v Plain TeXu. Pak své znalosti využijeme k vytvoření zvláštních triků.

MakeIndex napsal původně Pehong Chen a později byl program modifikován řadou programátorů. Původní verze byla přirozeně anglická a dodatečně byla implementována němčina. Od letošního pod-