Zpravodaj

Československého
sdružení uživatelů TEXu

4
92

bulletin

OBSAH

Přednovoroční

Jiří Veselý

Co napsat jako poslední ohlédnutí před nástupem do nového roku? Začnu vysvětlením a omluvou: výbor CSTUGu se nevyměnil na konci minulého roku, ale teprve teď a dosti nakvap. Písemně jsme žádné návrhy na nové funkcionáře nedostali, a tak jsme se snažili získat na poslední chvíli nové tváře do výboru. Neobešlo se to bez chyb: s některými kandidáty jsme nestačili promluvit, jiné jsme nestihli včas v dobrém „pomluvit“ — tedy představit je řádně voličům. Dokonce jsme zkomolili jedno jméno na kandidátce, šlo však o včas neobjevený překlep (kol. Bušovi se omlouváme).

Na valné shromáždění přišlo relativně dost lidí, a tak bylo s kým a před kým i trošku bilancovat. Loni jsme se k prodloužení mandátu výboru o jeden rok rozhodli po přijetí nabídky uspořádat v Praze EuroTEX 92; nebylo by totiž zcela fér nabídku přijmout a pak nechat jiné se dřít. A tak se starý výbor snažil se ctí vyrovnat s problémem, jak cizincům ukázat, že na takové věci u nás také stačíme, byť jsou naše podmínky trošku jiné (některé horší, ale některé i lepší).

Myslím, že za odstupující výbor mohu s malinkou stopou pýchy napsat: ano, povedlo se to. Znovu alespoň anonymně děkuji všem, kteří se o to zasloužili. O úspěšnosti svědčí řada děkovných e-mailů, dobré reference v zahraničních TEXových materiálech, zájem o sborník z konference i spokojenost většiny cizích i našich účastníků. Program probíhal bez zmatků (ani to nebývá zcela samozřejmé), jen s minimálními změnami a měl velmi pěknou úroveň. Společenská část byla přijata velmi pozitivně a mnoho zahraničních účastníků si odvezlo nezapomenutelné zážitky. I zahraniční účast byla velmi pěkná. Díky podpoře GUTenbergu a DANTE e.V. se mohla zúčastnit i řada zájemců z Ruska, Polska, Maďarska, Bulharska a Rumunska a prakticky poprvé se seznámit se světem TEXu za humny a kousek dál. A vzhledem k tomu, že nakonec přijelo přeci jen dost zájemců i ze západních zemí, mohli jsme z výdělku přispět na prodloužení jejich pobytu k účasti na tutoriálech i my. Celková bilance hospodaření je velmi pěkná, navzdory hospodářské situaci jsme nejen nezvýšili příspěvky, ale mohli jsme je v případě kolektivních členů i snížit na 4 000 Kčs ročně. Pro zajímavost: celkový příspěvek našich účastníků

zůstal pod 20 000 Kčs, vydělali jsme však více než desetinásobek této částky.

Co s ostatními věcmi obecného zájmu? Vlastníme korektor pro práci ve třech jazycích, před podepsáním je dohoda o koupi CSEDu (na naše přání bude možné provádět kontrolu korektorem i **uvnitř** tohoto editoru a tuto úpravu již Pavel Ševeček provedl). Pracuje se intenzivně na novém TEXovém **balíku**, který bude sdružením šířen. Budete mít možnost pracovat v pohodlnějším a konfigurovatelném prostředí (autorem nového menu je Petr Olšák). Díky svědomité přípravě jednání Olinem Ulrychem došla expertní skupina vcelku nekonfliktně k dohodám o obsahu i realizaci. Teď už se pilně pracuje, skupina (M. Bílý, M. Dont, K. Horák, P. Olšák, P. Novotný, O. Ulrych, M. Voců) si rozdělila úkoly a do konce ledna by mělo být vše pohromadě. Kupujeme větší množství disket (HD 5,25″), na kterých budeme nový balík kolektivním členům distribuovat.

Novému výboru do vínku předáme některé rozdělané věci i plány. Nedokázali jsme se dohodnout o strategii pro případný nákup **ChiTEXu**, jehož autorem je kol. Božovský. Rozhodnutí bude na novém výboru. K tomu jednu obecnější poznámku: celá světová TEXová komunita je z podstatné části založena na dobrovolné zájmové práci ve prospěch uživatelů tohoto typografického jazyka, snažíme se tedy podporovat projekty vedoucí k jeho zdokonalování. Skutečnou tržní hodnotu vzniklých produktů zpravidla není schopno ani sdružení zaplatit, poskytujeme prakticky příspěvky či stipendia pro rozvoj obecně potřebných věcí. Není to jen otázka zmíněného konvertoru, který by pro některé kolektivní členy byl přínosem, byť v jádru „odvádí od TEXu“; na valném shromáždění jsme se mohli seznámit v příspěvku kol. Kozlovského s jakýmsi interaktivním „Metafontovým editorem“. Jde o kolektivní produkt studentů MFF UK, který vzbudil pozornost i na EuroTEXu. Účastníci valného shromáždění projevili živý zájem **zejména** o tento směr jeho vývoje, byť toho bylo předvedeno více. Vzniká přirozená otázka, jak najít kompromis mezi koupí objednaného produktu na straně jedné a (někdy i dodatečnou) podporou produktu nabídnutého k obecnému využívání, často i zdarma. Takových nabídek zatím nebylo mnoho, nějaké zkušenosti jsme však již získali a přicházejí další (poslední nabídka ze Slovenska: TEXově orientovaný editor se zajímavými schopnostmi — projevili jsme zájem o testovací verzi). Citlivé řešení těchto věcí zbude na nastupující výbor.

Díky iniciativě Martina Bílého běží elektronický diskusní list v našich národních jazycích (`bez hacku a carek se lisi jazyky jeste`

`mene, tudy ale asi cesta nevede`). V tomto čísle by měla být i informace o získávání TEXwaru prostřednictvím archivu, který Martin bude provozovat a udržovat.

Probíhá jednání s vydavatelstvím Addison-Wesley (Germany) o vydání překladu knížky Norberta Schwarze o PlainTEXu („Introduction to TEX“). V souvislosti s tím bylo na valném shromáždění rozhodnuto vypsat konkurs na možného partnera pro vydání této knížky (a případně dalších). S tím souvisí i budoucí strategie sdružení v oblasti vydávání příruček apod. Prosíme, upozorněte na podmínky konkursu své známé:

> **Parametry:** Náklad překladu cca 2 000 ks, rozsah cca 280 str. formátu B5, obálka alespoň polokartón (rozhoduje praktičnost a nízká cena), zpracování z elektronické formy nebo z camera-ready předlohy. Nabídka by měla obsahovat cenu pro sdružení (cca 500 ks) a cenu pro veřejnost, popis distribučních možností, další relevantní údaje apod.

Dle zatímních jednání orientační cena práv pro překlad činí 7 % prodejní ceny, proto hledáme partnera, který ocení dobrou spolupráci a nebude chtít na publikaci příliš vydělávat. Překlad zajistíme. Nejlepším řešením je patrně dosažení rozumně nízké ceny pro veřejnost. Nabídku je třeba poslat doporučeně na adresu Dr. Karel Horák, MÚ ČSAV, Žitná 25, 115 67 Praha 1, do konce února 1993. Chtěli bychom vydat i několik dalších menších publikací o zajímavých TEXových formátech.

Dost už bylo všeho možného, i úvah o budoucím uspořádání ČSTUGu. Přáním výboru i účastníků valného shromáždění je dle možnosti na dosavadním uspořádání nic neměnit. Budoucnost ukáže, nakolik jsou tato přání reálná, osobně věřím, že vše půjde dobře zvládnout.

Za osud naší organizace v příštím roce bude zodpovědný staronový výbor (některé posily s největším obdrženým počtem hlasů jsme kooptovali), který bude pracovat ve složení:

předseda: Karel Horák

členové: Martin Bílý, Miroslav Dont, Janka Chlebíková, Ladislav Lhotka, Karol Nemoga, Petr Novotný, Petr Olšák, Štefan Porubský, Pavel Sekanina, Petr Sojka, Oldřich Ulrych, Jiří Veselý, Zdeněk Wágner, Jiří Zlatuška (kolega Petr Demel nemohl funkci pro časové zaneprázdnění přijmout).

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:

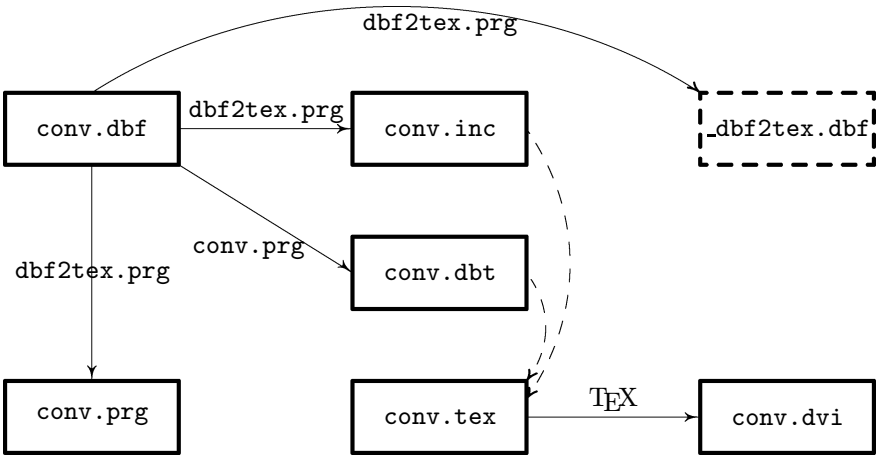
1. Možnost zjistit název zpracovávané databáze (název vlastního souboru bez extenze).
2. Možnost zjistit strukturu databáze (počet polí, jejich typ a názvy).
3. Výstup textu do souboru (a to včetně některých speciálních znaků — jestliže například znak " je vyhrazen v databázi pro vyznačování

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-

zimu již existuje česká a slovenská verze pod názvem ***C*sIndex**. Tato verze je k dispozici v Československém sdružení uživatelů TEXu a v archívu `cs.felk.cvut.cs` v adresáři `[pub.tex.makeindex2-11]`. Soubor se jmenuje `csidx2_11.zip`. Alternativně můžete použít ftp na `vax.felk.cvut.cs`.

Zde si dovolím drobnou poznámku. Pokud se rozhodnete získat program ***C*sIndex**, pak již nepotřebujete původní ***MakeIndex***, soubor `csidx2_11.zip` totiž obsahuje kompletní distribuci programu ***MakeIndex*** verze 2.11 a navíc podporu češtiny a slovenštiny. V ***C*sIndex**u se navíc dá čeština/slovenština vypnout, takže program pak pracuje zcela identicky jako ***MakeIndex***.

Čím je čeština a slovenština specifická?

Nejprve si vysvětlíme, proč vlastně potřebujeme speciální verzi programu ***MakeIndex*** pro češtinu a slovenštinu. Zdánlivě by bylo jednodušší, kdybychom vytvořili nějaký filtr a využili schopností programu ***MakeIndex***. Zdání ovšem klame a v počítačovém světě to často platí úplně stejně.

Německé třídění nečiní žádné zvláštní obtíže. Přehlasovaná písmena patří v abecedě na stejná místa jako příslušná písmena nepřehlasovaná, takže ve slovníku můžeme najít slova v pořadí:

$$\text{braun} \prec \text{bräunen} \prec \text{Braunkohle} \prec \text{bräunlich}.$$

Problémy použití ***MakeIndex***u ve spojení s němčinou jsou jinde, ale k tomu se dostaneme později.

V češtině a slovenštině je situace složitější. Pro řadu písmen platí obdobná pravidla jako pro německé přehlásky. V češtině (a zřejmě i slovenštině) jsou ovšem háčky a čárky významotvorné. Nemůžeme tudíž sdělit ***C*sIndex**u, že **a** i **á** jsou identické. V takovém případě bychom nedostali správné:

$$\text{car} \prec \text{cár} \prec \text{carevna}.$$

Rozdíl mezi krátkým a dlouhým písmenem se totiž ignoruje, ale bere se na něj ohled, jsou-li slova jinak identická. Primární řadicí platnost však mají **č**, **ř**, **š** a **ž**. Jinak by se totiž správné pořadí

$$\text{zlato} \prec \text{zlý} \prec \text{žluna} \prec \text{žlutý}$$

změnilo na obskurní

$$\text{zlato} \prec \text{žluna} \prec \text{žlutý} \prec \text{zlý}.$$

Samostatnou kapitolu tvoří dvojhláska **ch**. Ta má také primární řadicí platnost a patří mezi **h** a **i**. Známe ovšem složená slova, kde **c** a **h** není **ch**. Přitom neexistuje uzavřený algoritmus na rozeznávání, zda se v daném případě jedná o dvojhlásku nebo dvě samostatná písmena. Vezměme si například slovo *mochnatý* odvozený od slova mochna, a *mochnátý*, znamenající „mající mnoho končetin“. Přitom se opticky obě slova liší pouze jednou čárkou. Ještě složitější je případ zkratek. Kdokoliv si může vymyslet, že **C**vičná **H**orská **CH**ata bude mít zkratku **CHCH**. Zde všechny algoritmy selhávají úplně a zbývá jen lidský rozum. Využíváme zde toho, že TEX prázdné složené závorky `{}` interpretuje jako nic. Můžeme je proto s výhodou použít k oddělení **C** a **H** v případech, kdy netvoří dvojhlásku. Index pro CHCH tedy vytvoříme jako `\index{C{}HCH}`.

*Co **C**s**Index** umí a co neumí?*

V dokumentaci k české/slovenské verzi programu si můžete přečíst, s jakým záměrem byl ***C**s**Index*** vytvořen. Hlavním cílem totiž bylo vytvoření funkčního programu s vynaložením minimálního úsilí. Proto není funkce programu zdaleka optimální a nejsou implementovány některé funkce, které by mohly být užitečné.

Program tedy umí „pouze“ řadit česká a slovenská slova podle požadavků normy ČSN 01 0181. Přitom všechna písmena musí být kódována v jednom ze tří kódů — KOI 8 ČS, Latin 2 nebo v kódu Kamenických. Zápisu `\'a` či `\v{z}` ***C**s**Index*** nerozumí. Stejně tak nepřijme přehlasovaná písmena kódovaná pomocí `\"u` apod. Zde je však pomoc snadná. Program `dvi2dvi`, který distribuuje Československé sdružení uživatelů TEXu, dokáže tato písmena zadaná přímo z klávesnice převést tak, aby se správně vytiskla i se sedmibitovými fonty computer modern.[1]) Ve spojení s češtinou a slovenštinou není ale implementována komprese mezer. Na to musí pamatovat uživatel při specifikaci svých požadavků pro ***C**s**Index***.

Na závěr této kapitoly ještě zdůrazníme, že ***C**s**Index*** je vyvinut pro češtinu a slovenštinu. Rozumí tedy písmenům obvyklým v těchto jazycích, ale nedokáže si poradit s francouzským ç, dánským ø a podobnými znaky.

[1]) Osmibitové dc fonty jsem pro nedostatek místa na svém disku nezkoušel.

Jak se definují makra

Účelem tohoto článku je, aby jej pochopili i laici. Lze ovšem předpokládat, že laici používající zejména LaTeX nemusí být schopni psát složitější makra. Pokud se někdo cítí být tímto soudem podceněn, nechť přijme autorovu omluvu a další řádky přeskočí.

Někomu by se mohl zdát následující výklad příliš rozsáhlý a odtržený od problémů indexace. Zde se ale musíme rozhodnout, co vlastně chceme. Pokud se někdo smíří s tím, že bude dělat pouze velmi jednoduché rejstříky, pak nemusí nic vědět a může prostě používat to, co už je v LaTeXu uděláno. Chce-li však někdo plně využívat možnosti, které ***MakeIndex*** poskytuje, pak musí umět psát složitá makra, nebo je alespoň musí umět upravovat pro své účely. Protože by tento článek měl přinést některé recepty, musíme nejprve vysvětlit stavební kameny, které budeme později používat.

Jak již bylo řečeno, ***MakeIndex*** byl napsán pro LaTeX. Měli bychom se tudíž zabývat psaním LaTeXových maker. Pro nás však bude zajímavý též Plain TeX, a to ze dvou důvodů. Jednak bylo v úvodu slíbeno, že vysvětlíme tvorbu rejstříku v Plain TeXu, ale navíc jsou případy, kdy definice maker prostředky LaTeXu je neschůdná nebo zcela nemožná.

Nejprve si vysvětlíme, jak napsat velmi jednoduché makro. Nazveme jej `\simplemacro` a jeho funkcí bude pouze napsání neproměnného textu.[2]) V LaTeXu takové makro nadefinujeme pomocí příkazu

```
1 \newcommand{\simplemacro}{Toto je velmi jednoduché makro}
```

V Plain TeXu je syntaxe velmi podobná:

```
2 \def\simplemacro{Toto je velmi jednoduché makro}
```

Před okamžikem jsme řekli, že makro se v textu nahradí neproměnným textem. To je sice pravda, ale pouze částečná. Makro totiž můžeme kdykoliv předefinovat. V LaTeXu k tomu slouží příkaz `\renewcommand`:

```
3 \renewcommand{\simplemacro}{Nyní zde máme změněnou definici}
```

Plain TeX žádný speciální příkaz nepotřebuje, protože příkaz `\def` nekontroluje, zda je již makro definováno.

Výše uvedené jednoduché makro je ovšem málo flexibilní. Často potřebujeme do makra přenést proměnnou informaci, kterou je třeba nějak interpretovat. Takovou činnost provádějí makra s parametry. Minimální

[2]) Tento dokument je psán s využitím stylu „DOC.STY“ vytvořeného Frankem Mittelbachem. Tento styl automaticky čísluje řádky a zajišťuje indexaci všech maker.

počet parametrů je přirozeně 1, maximální počet parametrů je 9. Je to dáno tím, že se v makru odvoláváme na parametr znakem `#`, za nímž následuje pořadové číslo parametru. Makro se dvěma parametry definujeme v LaTeXu

```
4 \newcommand{\params}[2]{První parametr: #1, druhý parametr: #2}
```

V Plain TeXu je specifikace počtu parametrů složitější. Důvod této složitosti si objasníme později, ale napřed se podívejme na ekvivalent výše uvedeného LaTeXového příkazu

```
5 \def\params#1#2{První parametr: #1, druhý parametr: #2}
```

V obou případech je jako parametr převzat pouze jeden token.[3]) Makro `\params` tedy musíme volat:

```
6 \params{první slovo}{druhé slovo}
```

Za okamžik se dostaneme do situace, kterou nelze řešit v LaTeXu. Budeme chtít, aby makro `\params` fungovalo stejně jako v předchozí definici, ale s tím rozdílem, že se jako první parametr vezme řetězec až do znaku čárka, a druhým parametrem bude řetězec až do znaku tečka. Tedy předchozí volání makra chceme nahradit pohodlnějším

```
7 \params první slovo,druhé slovo.
```

K tomu účelu trochu upravíme výše uvedenou definici:

```
8 \def\params#1,#2.{První parametr: #1, druhý parametr: #2}
```

Všimněte si čárky a tečky oddělující parametry. Zdánlivou větší složitostí definice platíme za větší možnosti, které nám ovšem vydatně pomohou při řešení složitějších problémů.

Definice prostředí

Prostředí (environment) je LaTeXový objekt, který v Plain TeXu nemá obdobu. Text uvnitř prostředí má určité vlastnosti, které mimo něj nemá. Každý LaTeXista zná prostředí ⟨*minipage*⟩, ⟨*tabular*⟩, ⟨*figure*⟩, ⟨*verbatim*⟩ a řadu jiných. Uživatel si ale může nadefinovat své vlastní prostředí. Podobně jako u definice maker, `\newenvironment` slouží k definici nového prostředí a `\renewenvironment` k předefinování již definovaného prostředí.

[3]) Přesnou definici, co je to „token“, uvádí TeXbook. Zde jen zjednodušeně uvedeme, že token je buď jeden znak, nebo jedno makro (např. `\params`), nebo libovolný řetězec uzavřený ve složených závorkách.

Příklad definice prostředí si předvedeme na jednoduchém prostředí pro tisk seznamu literatury. Podobné prostředí jsem před časem definoval pro styl určený k přípravě rukopisů pro *Fluid Phase Equilibria*. Základním požadavkem je tisk menším typem písma, přičemž první řádek citace začíná od kraje a ostatní řádky jsou odsazeny o `\parindent`. Zde je příslušná definice. Nejprve do prvního páru složených závorek vložíme příkazy, které se provedou jako rozvinutí příkazu `\begin{biblio}`.

```
\newenvironment{biblio}{\subsection*{Literatura --- demonstrace}
  \rm\small
  \def\cite{\par\hang\noindent}
}%
```

Do druhého páru složených závorek zapíšeme příkazy, které se mají provést při rozvinutí `\end{biblio}`:

```
{\vspace{2ex}\par}
```

Poslední řádek první části definice prostředí ⟨*biblio*⟩ je ukončen komentářovým znakem %. Bez tohoto znaku by LaTeX byl zmaten mezerami a `\end{biblio}` by způsobil chybu „Use of `\endbiblio` doesn't match its definition...“. Pokud nevíte, kde je procento nutné, můžete pro jistotu ukončovat procentem všechny řádky při definici všech maker. V žádném případě tím nemůžete nic zkazit.

Literatura — demonstrace

Nyní se můžeme podívat na výsledek své definice. Mohl bych sice vymyslet nějaké citace, ale základní vlastnosti takto jednoduchého prostředí lze dokumentovat i na obyčejném textu.

Všimněte si obráceného odsazování odstavců, které je způsobeno makrem `\hang`. Každý odstavec tedy musí začínat příkazem `\cite`.

Příkaz `\begin{biblio}` za nás napsal název (tj. Literatura — demonstrace), přepnul font a definoval makro `\cite`. Zde jsme příkaz pro tisk nadpisu změnili na `\subsection*`, aby nerušil svým vzhledem posloupnost výkladu, a definici jsme maximálně zjednodušili.

Při rozvoji `\end{biblio}` se uzavře odstavec. Kdybychom totiž před nebo za `\end{biblio}` zapomněli udělat prázdný řádek, LaTeX by začal dělat podivné věci. A právě proto končí definice prostředí ⟨*biblio*⟩ příkazem `\par`.

V předchozí kapitole jsme řekli, že existují komplikovaná makra, která v LaTeXu lze definovat jen obtížně nebo dokonce vůbec ne. Vzniklé potíže nám pak vyřeší Plain TeX. Totéž lze říci i o prostředích. Chceme-li

tudíž definovat složitá prostředí, nebo alespoň pochopit, jak takové prostředí nadefinoval nějaký „čaroděj“, abychom si je mohli upravit podle vlastních potřeb, musíme si říci něco o tom, co vlastně LaTeXdělá, když narazí na příkazy `\begin{envir}` a `\end{envir}`. Činnost je jednoduchá. Místo `\begin{envir}` LaTeX v principu provede

```
14 \begingroup \envir
```

a místo `\end{envir}`

```
15 \endenvir \endgroup
```

Úmyslně jsem tu nerozepisoval jisté formální kontroly, které LaTeX provádí a které obyčejné uživatele nezajímají.

Z výše uvedeného plyne, že nikde nemusíte najít řídící slovo `\newenvironment{biblio}`, přestože `\begin{biblio}` a `\end{biblio}` funguje. Místo toho jsou definována makra `\biblio` a `\endbiblio`.

Pro LaTeXisty je asi ještě nutno vysvětlit, co příkazy `\begingroup` a `\endgroup` znamenají. Tyto příkazy mají význam složených závorek[4]) a způsobují, že definice provedené uvnitř jsou lokální a vně těchto příkazů nejsou definovány. V prostředí ⟨*biblio*⟩ jsme si nadefinovali makro `\cite`. Pokud bylo stejnojmenné makro definováno před příkazem `\begin{biblio}`, pak mu TeX po `\end{biblio}` vrátí původní význam. Jinak nebude vně prostředí toto makro definováno.

Podrobnosti k definici maker

V předchozí kapitole jsme viděli, že makra definovaná uvnitř prostředí jsou v tomto prostředí lokální. To se nám ale někdy nehodí a chtěli bychom definici makra exportovat ven. K tomuto účelu TeX nabízí globální definice, což zařizuje příkaz `\global`. Globální definice jsou ovšem dosti časté. Proto TeX zná příkaz `\gdef`, který je zkratkou místo `\global\def`.

Někdy nám záleží na tom, ve kterém okamžiku dojde k rozvinutí makra. Vezměme si následující příklad:

```
16 \def\slovo{cokoliv}
17 \def\macro{Napiš \slovo.}
18 \def\slovo{něco jiného}
19 \macro
```

[4]) Vysvětlení, proč tam nemohou být složené závorky, přesahuje rámec tohoto textu. Pokud nevěříte, zkuste si to a uvidíte, jak bude TeX zběsile nadávat.

Vyzkoušejte si, že tyto příkazy vytisknou „Napiš něco jiného.“Je to proto, že makro `\slovo` se rozvine až v okamžiku použití makra `\macro`. Pokud bychom chtěli rozvinout `\slovo` již v okamžiku definice makra `\macro`, museli bychom použít `\edef` místo `\def`. To je další případ, který nelze řešit prostředky LaTeXu. Příkazy

```
\def\slovo{cokoliv}
\edef\macro{Napiš \slovo.}
\def\slovo{něco jiného}
\macro
```

potom vytisknou „Napiš cokoliv.“

I tento příkaz má globální verzi. Místo `\global\edef` stačí psát `\xdef`.

Jak LaTeX vytváří rejstřík

Po dlouhém úvodu se konečně dostáváme k meritu věci. Vysvětlíme si, kde se vlastně bere informace pro tvorbu rejstříku.

Nejprve je nutno LaTeXu říci, že chceme vytvářet rejstřík. Proto musíme načíst `makeidx.sty` a hned na začátek dokumentu uvést `\makeindex`. První řádky našeho dokumentu tedy budou:

```
\documentstyle[...,makeidx,...]{...}
\makeindex
```

Tečky v závorkách označují, že můžeme současně specifikovat libovolné další „style options“ a můžeme použít libovolný „style“, pokud to dává smysl.

Jednotlivá slova nebo výrazy potom vkládáme do rejstříku makrem `\index`. Jediným parametrem tohoto makra je výraz, který chceme do rejstříku vložit, číslo stránky se doplní automaticky. Jako příklad použití makra `\index` uvedeme větu:

```
Jedním z nejdokonalejších DTP\index{DTP} programů
je beze sporu TeX.
```

Všimněte si, že DTP se v této větě vyskytuje dvakrát, jednou jako část věty, jednou jako parametr makra `\index`. Makro `\index` totiž neprovádí žádný viditelný výstup. Slouží pouze k zápisu do rejstříku.

Nakonec dokumentu, do místa, kde se má rejstřík vytisknout, napíšeme `\printindex`. Tím je LaTeXová část skončena.

Teď se přirozeně naskýtá otázka, kdy a kde se rejstřík setřídí podle abecedy. Při vysvětlování se vrátíme o několik odstavců zpět. V souboru `makeidx.sty` je definice makra `\printindex`. Makro `\makeindex` otevře výstupní soubor s příponou `.idx` a zajistí správnou definici makra `\index`. Původně je totiž toto makro definováno tak, aby nedělalo nic. Tím se zabrání tomu, aby se omylem zapisovalo do neexistujícího souboru. Soubor se záznamy vytvořenými voláním makra `\index` se potom zpracuje programem ***MakeIndex*** nebo, pokud obsahuje češtinu či slovenštinu, programem ***CsIndex***. Výsledkem je soubor s příponou `.ind`, který načte a vytiskne makro `\printindex`. A tím je práce hotova.

Funkce indexových maker

Již dříve jsme uvedli, že v LaTeXu je již všechno uděláno. Uživatel tedy nemusí příliš rozumět, jak indexová makra pracují. Následující text je tedy určen zejména pro toho, kdo používá Plain TeX a chtěl by vytvářet rejstřík. Závěr této kapitoly však bude zajímat i pokročilejší LaTeXisty.

Všechna makra, uvedená v této kapitole, jsou převzata ze souboru `latex.tex`, LaTeX Version 2.09 ⟨9 Jan 1990⟩.

V předchozí kapitole jsme uvedli, že makro `\makeindex` otevře soubor s příponou `.idx` a předefinuje makro `\index`. Nyní se podíváme, jak se to ve skutečnosti provádí.

```
\def\makeindex{if@filesw \newwrite\@indexfile
  \immediate\openout\@indexfile=\jobname.idx
  \def\index{\@bsphack\begingroup
    \def\protect####1{\string####1\space}\@sanitize
    \@wrindex}
  \typeout{Writing index file \jobname.idx }
  \fi}
```

Makro je na první pohled složité, proto se u něj trochu zastavíme. První zmínka se týká znaku ‚@‘. Pokud do svého dokumentu napíšete například `\@indexfile`, TeX vám napíše nepříliš srozumitelnou zprávu „Illegal use of `\spacefactor`“. Příčina spočívá v tom, že ‚@‘ není písmeno. V souborech „style option“, tj. v těch, které uvádíme v hranatých závorkách příkazu `\documentstyle`, je změněna kategorie tohoto znaku na 11, což je písmeno. Důvodem této „složitosti“ je snaha zabránit tomu, aby uživatel omylem nepoužil makro, které pro něj není určeno a mohlo by

důkladně zhroutit jeho dokument. Chcete-li podobnou techniku použít v PLAIN TEXu, uveďte před inkriminované příkazy

```
\catcode`\@=11
```

a potom nezapomeňte vše vrátit do původního stavu příkazem

```
\catcode`\@=12
```

TEXpert může oprávněně namítnout, že výše uvedený postup není zcela správný. Jenže, pokud nerozumíte příkazům `\catcode`, pak se nedostanete do situace, kdy by výše uvedená jednoduchá makra způsobila chybu, a proto podrobnou diskusi ponecháme do jiného článku, který se nebude zabývat vytvářením rejstříku.

Po nezbytné odbočce se tedy vrátíme k makru `\makeindex`. LATEX umožňuje zakázat jediným přepínačem tvorbu všech pracovních souborů. Používá se to pro závěrečný překlad dokumentu, kdy jsou všechny indexové soubory, pomocný soubor s křížovými odkazy, soubor pro obsah a seznam obrázků a tabulek již vytvořeny. Překlad dokumentu se tím výrazně urychlí. Rozhodování provádí přepínač `\if@filesw` a rozhodovací blok je uzavřen příkazem `\fi` na konci definice.

Makro `\jobname` obsahuje název hlavního souboru, který zpracováváte TEXem. Z tohoto jména je odstraněna přípona. V okamžiku psaní tohoto článku nemohu říci, jak jej v poslední fázi upraví redakce, a tudíž nevím, co bude `\jobname` ve finální fázi obsahovat. Proto pro demonstraci napíši: `\jobname` = tug4.

Makro `\protect` je důvěrně známo všem LATEXistům. Toto makro dočasně zabraňuje expanzi makra bezprostředně následujícího a jeho definice se mění podle okolností.

Vlastní zápis provádí makro `\@wrindex`. Na první pohled vypadá složitě, ale myslím si, že nepotřebuje komentář. Snad jen pro uživatele PLAIN TEXu uvedu, že `\thepage` obsahuje číslo stránky v té formě (roman, arabic, atd.), jak bude v dokumentu vytištěno.

```
\def\@wrindex#1{\let\thepage\relax
  \xdef\@gtempa{\write\@indexfile{\string\indexentry{#1}
    {\thepage}}}
  \endgroup\@gtempa
  \if@nobreak \ifvmode\nobreak\fi\fi\esp@hack}
```

Pokud chceme použít `\@fileswfalse` pro zákaz zápisu do indexového souboru, musíme si nadefinovat makro `\index`.

```
\def\index{\@bsphack\begingroup \@sanitize\@index}
```

```
\def\@index#1{\endgroup\@esphack}
```

Samozřejmě nesmíme zapomenout na přepínač, který jsme použili v předchozích makrech.

```
\newif\if@filesw \@fileswtrue
```

V předchozích definicích se vyskytují makra, o nichž jsme si dosud nic neřekli. Jedním z nich je `\@sanitize`. Účelem tohoto makra je deaktivace všech speciálních znaků s výjimkou složených závorek, aby argument makra `\index` nezpůsobil nežádoucí efekty. Všimněte si, že v definici `\@sanitize` nejsou žádné mezery a všechny řádky s výjimkou posledního jsou ukončeny komentářovým procentem. Mezery v definici i na koncích řádků by totiž způsobily dokonalé zmatení TEXu.

```
\def\@sanitize{\@makeother\\@makeother\\\@makeother\$%
\@makeother\&\@makeother\#\@makeother\^\@makeother\^^K%
\@makeother\_\@makeother\^^A\@makeother\ \@makeother\~}
\def\@makeother#1{\catcode`#112\relax}
```

Je přirozené, že makro, které dělá tak brutální zásahy, smíme použít pouze uvnitř „grupy“, aby se vše vrátilo do původního stavu. Proto je hned v úvodu makra `\index` uveden `\begingroup` a `\@wrindex` i `\@index` obsahují `\endgroup`.

Makra `\@bsphack` a `\@esphack` se používají v případě, kdy naše makro pouze provádí nějakou činnost a neprodukuje žádný viditelný výstup. Těmito makry pak zajistíme, že případné mezery v definici našeho makra neznehodnotí formátování.

```
\newdimen\@savsk
\newcount\@savsf
\def\@bsphack{\relax\ifmmode\else\@savsk\lastskip
  \ifhmode\@savsf\spacefactor\fi\fi}
\def\@esphack{\relax\ifmmode\else\ifhmode\spacefactor\@savsf
  {}\ifdim\@savsk >\z@ \global\@ignoretrue \ignorespaces
  \fi\fi\fi}
```

Zde `\z@` je `\count` obsahující hodnotu nula. Tento zápis je pro TEX rychlejší než uvedení znaku ‚0‘.

V souboru `makeidx.sty` je nadefinováno makro `\printindex`, které vytiskne setříděný rejstřík. Příslušná definice zní

```
\def\printindex{\@input{\jobname.ind}}
```

Makro `\@input` načte soubor pouze v případě, že daný soubor existuje. V opačném případě zobrazí varování. Nečeká tedy na zadání jiného jména souboru, jako to dělá `\input`.

```
\def\@input#1{\openin1 #1
  \ifeof1 \typeout{No file #1.}
  \else\closein1 \relax\input #1 \fi}
```

V LaTeXu je na posledním řádku předchozí definice `\@@input` místo makra `\input`. Makro `\input` je totiž v LaTeXu předefinováno a původní definice z Plain TeXu je schována právě v `\@@input`.

A kde je tedy definice hlavičky, která je nadepsána na začátku rejstříku? Odpověď je jednoduchá. Na začátku souboru `\jobname.ind` je `\begin{theindex}` a na jeho konci je`\end{theindex}`. Na uživateli tedy je, aby si nadefinoval prostředí ⟨*theindex*⟩. Makra pro tisk tohoto článku jsou poměrně komplikovaná (jsou převzata z „Mainz Distribution“), takže na následujících řádcích budu úmyslně trochu lhát. Prostředí pro tisk rejstříku maker je v zásadě definováno:

```
\renewenvironment{theindex}
 {\begin{multicols}{3}[\section*{Rejstřík maker}
  Čísla vytištěná italikou označují stránky, kde je
  příslušné makro popsáno. Podtržená čísla odkazují na definici
  a všechna ostatní ukazují místo, kde je makro použito.]
 \IndexParms \ignorespaces}%
```

Zde jsme si připravili tisk ve třech sloupcích. Prostředí ⟨*multicols*⟩ je definováno v `multicol.sty`, který je součástí „Mainz Distribution“. `\IndexParms` zde úmyslně nevysvětlíme. Později si ukážeme, že ***MakeIndex*** zapisuje do souboru `\jobname.ind` uživatelem specifikovaná makra. Jejich definice obsahuje potom `\IndexParms`.

Druhá část definice prostředí už jenom ukončí třísloupcový tisk.

```
{\end{multicols}}
```

Používáte-li Plain TeX, musíte nadefinovat slova `\theindex` a `\endtheindex` a provést odpovídající změny v souborech pro ***MakeIndex***.

Uživatelské definice pro *MakeIndex*

LaTeXumožňuje specifikovat v příkazu `\documentstyle` jména speciálních souborů maker, která definují uživatelský styl. Podobnou možnost má i ***MakeIndex***. Nebudeme se zde zabývat podrobným popisem jednotlivých parametrů. Spokojíme se pouze s konstatováním, že pro Plain TeXmusíme uvést

```
preamble    "\\theindex\n"
postamble   "\n\n\\endtheindex\n"
```

Pokud bychom tyto parametry neuvedli, platilo by standardně:

```
preamble    "\\begin{theindex}\n"
postamble   "\n\n\\end{theindex}\n"
```

Tato verze je vhodná pro LaTeX, ale jak jsme si vysvětlili dříve, pro Plain TeX se použít nedá.

Triky s rejstříkem

Teď už umíme dost na to, abychom si mohli ukázat nějaké triky. Vezměme si případ, že chceme vytisknout sborník konference, která má více odborných sekcí označených písmeny, a v každé sekci se přednášky číslují od jedničky. Kromě toho chceme vytisknout autorský rejstřík a seznam klíčových slov. Odkaz v rejstříku nemá být stránka, ale označení přednášky.

Předpokládejme, že nějaká databáze nám připraví ASCII soubor vhodný pro zpracování TeXem. Označení začátku sekce nechť provede makro `\newsect` a pro každou přednášku se vytvoří série příkazů:

```
\startref
\aut{Markalous J. K.}
\aut*{Babočka A.}
\tit{Studium toku písku v~přesýpacích hodinách}
\kwd{hodiny!přesýpací,písek}
```

Příkazem `\aut*` jsme označili hlavního autora. Znak ‚!‘ v argumentu makra `kwd`, jež definuje klíčové slovo, vytváří hierarchický index. V další přednášce budeme mít „hodiny!kyvadlové“.

Co si ale počneme, když budeme chtít vložit vykřičník jako součást klíčového slova? ***MakeIndex*** nám naštěstí nabízí řešení. Stačí použít uvozovky, které z následujícího znaku dělají znak obyčejný. Napíšeme-li tedy `"!`, ztratí vykřičník svůj speciální význam a stane se obyčejným vykřičníkem. A zde je ten slíbený problém s němčinou. Tam se totiž `\"` používá pro označení přehlásek. Takový problém se dá řešit několika způsoby, a to buď v TEXu, nebo v ***MakeIndex***u. Nejsnadnější je asi předefinování ve stylovém souboru pro ***MakeIndex***, takže roli uvozovek převezme jiný znak.

Nyní se však již podíváme, jak budeme řešit nastolený problém.

Příprava indexových souborů

Než začneme s definicemi, uvedeme, že tento soubor je psán jako „style option“ pro LATEX. Jak bylo vysvětleno v kapitole o funkci indexových maker, znak ‚@‘ se v takovém případě považuje za písmeno a můžeme jej tedy použít ve jménech maker, která by uživatel neměl používat. Dále využijeme LATEXové čítače, které mají některé zajímavé vlastnosti a ušetří nám trochu práce.

Nejprve musíme připravit indexové soubory. Přitom se poučíme z makra `\makeindex`. Nesmíme však zapomenout, že indexové soubory jsou dva. Prozatím jim přidělíme symbolická jména `\aut@file@nm` a `\kwd@file@nm`.

```
\def\PrepareIndex{\newwrite\aut@file \newwrite\kwd@file
  \immediate\openout\aut@file=\aut@file@nm.idx
  \immediate\openout\kwd@file=\kwd@file@nm.idx
  \def\aut@index{\@bsphack\begingroup
    \def\protect####1{\string####1\space}\@sanitize
    \aut@wrindex}
  \def\kwd@index{\@bsphack\begingroup
    \def\protect####1{\string####1\space}\@sanitize
    \kwd@wrindex}
  \def\close@index{\immediate\closeout\aut@file
                   \immediate\closeout\kwd@file}
  \typeout{Writing author and keyword index files.}
  }
```

Makra pro zápis indexů

Při konstrukci maker se opět poučíme z LaTeXu, a to z makra `\@wrindex`. Nyní ale máme úmyslně makra dvě. Chceme zabránit tomu, aby uživatel omylem použil při třídění nesprávný stylový soubor. Proto místo standardního `\indexentry` budeme vytvářet `\authorentry` pro autorský rejstřík a `\keywordentry` pro rejstřík klíčových slov.

```
\def\aut@wrindex#1{
  \xdef\@gtempa{\immediate\write\aut@file
                {\string\authorentry{#1}{\therefnum}}}
  \endgroup\@gtempa
  \if@nobreak \ifvmode\nobreak\fi\fi\@esphack}
\def\kwd@wrindex#1{
  \xdef\@gtempa{\immediate\write\kwd@file
                {\string\keywordentry{#1}{\therefnum}}}
  \endgroup\@gtempa
  \if@nobreak \ifvmode\nobreak\fi\fi\@esphack}
```

Původní makro `\@wrindex` zapisovalo do indexu číslo stránky. To je ovšem bezpečně známo až v okamžiku, kdy se volá „`\output`“. Proto se zápis odkládá až do okamžiku zpracování stránky. V našem případě je číslo přednášky známo okamžitě, a proto budeme index okamžitě zapisovat do souboru.

Podobně jako ve standardním LaTeXu nadefinujeme nyní indexová makra, která nedělají nic. Navíc ještě vytvoříme nic nedělající makro `\close@index`.

```
\let\close@index\relax
\def\aut@index{\@bsphack\begingroup \@sanitize\@index}
\def\@index#1{\endgroup\@esphack}
\let\kwd@index\aut@index
```

Makra pro tisk rejstříků

Tisk rejstříků bude udělán účelově pouze pro tuto demonstraci. Příklady totiž budou krátké, takže by zde nevypadalo hezky, kdyby se každý rejstřík tiskl samostatně. Vytiskneme tedy oba rejstříky do dvou sloupců, které rozdělíme tak, jak to vyjde.

```
\def\indexes{
  \close@index
```

```
\begin{multicols}{2}[\hrule]
\@input{\aut@file@nm.ind}
\@input{\kwd@file@nm.ind}
\ifvmode\hbox{}\par\fi
\end{multicols}
\nobreak\hrule\par}
```

Zde je jedna z největších zrad, která způsobila opoždění tohoto příspěvku. Při počátečním překladu LaTeXem se totiž teprve vytvářejí soubory `idx`, takže soubory `ind` ještě vůbec nemohou existovat. Obsah prostředí ⟨*multicols*⟩ je tudíž prázdný. Kdyby se prázdné prostředí chovalo tak, že by prostě nic netisklo, bylo by vše jasné. Skutečnost je však mnohem drastičtější. V tomto případě se totiž z dokumentu ztratí beze stopy celá stránka. A hledejte to, když víte, že v dokumentu máte ještě řadu dalších nefungujících maker...

Dále si musíme nadefinovat prostředí pro tisk jednotlivých rejstříků.

```
\newenvironment{theautindex}{\subsection*{Autorský rejstřík}
 Číslo vytištěné tučně odkazuje na přednášku, kde je příslušný
 autor hlavním autorem.\vspace{2.5ex}\par\CsIndexParms
 \let\item\@idxitem \ignorespaces}%
 {}
\newenvironment{thekwdindex}%
{\subsection*{Klíčová slova}\CsIndexParms
\let\item\@idxitem \ignorespaces}%
{}
```

Definice makra `\CsIndexParms` je převzata z `doc.sty` od Franka Mittelbacha.

```
\def\CsIndexParms{%
  \parindent \z@
  \columnsep 15pt
  \parskip 0pt plus 1pt
  \rightskip 15pt
  \mathsurround \z@
  \parfillskip=-15pt
  \def\main##1{{\bf ##1}}
  \def\@idxitem{\par\hangindent 30pt}%
  \def\subitem{\@idxitem\hspace*{15pt}}%
  \def\subsubitem{\@idxitem\hspace*{25pt}}%
```

```
  \def\indexspace{\par\vspace{10pt plus 2pt minus 3pt}}%
}
```

Čítače

Budeme potřebovat dva čítače. Jeden bude označovat odborné sekce, druhý bude číslovat přednášky. Ten se musí automaticky vynulovat, když zahájíme novou sekci. K tomuto účelu použijeme LaTeXové definice.

```
\newcounter{sect}
\newcounter{refnum}[sect]
\setcounter{sect}{\z@}
\def\thesect{\Alph{sect}}
\def\therefnum{\Alph{sect}.\arabic{refnum}}
```

Zpracování záznamu

Makro `\newsect` vytiskne nadpis odborné sekce. Použijeme postup, který není z koncepčních důvodů příliš vhodný, ale dá nám nejméně práce.

```
\def\newsect#1{\stepcounter{sect}\subsection*{\thesect\ #1}}
```

Makro `\startref` provede jedinou činnost: vytiskne tučně číslo přednášky.

```
\def\startref{\def\@delim{}\stepcounter
                  {refnum}{\noindent\bf\therefnum\ }}
```

Makro `\aut` vypisuje postupně jména autorů. Zde je třeba zkontrolovat, zda se jedná o hlavního autora (tj. použili jsme `\aut*`). Pro tento test použijeme LaTeXový příkaz `\@ifstar`. Všimněte si, že zde ještě nebudeme rozebírat parametry. Tuto činnost svěříme dalším makrům.

```
\def\aut{\@ifstar{\main@aut}{\@aut}}
```

Obě makra, `\main@aut` i `\@aut` provádějí v podstatě totéž. Pouze `\main@aut` navíc označí autora jako hlavního tím, že před jeho jméno vytiskne hvězdičku a označí vhodným způsobem výstup pro ***CsIndex***. Proto se v obou případech po krátké přípravě zavolá makro `\@@aut`.

```
\def\main@aut{\@delim\hbox{$\ast$}\def\ix@style{|main}\@@aut}
\def\@aut{\@delim\def\ix@style{}\@@aut}
```

Makro `\@@aut` předefinuje význam pomocného makra `\@delim`, zavolá indexové makro a zapíše jméno autora do výstupního souboru.

```
\def\@@aut#1{\def\@delim{, }\aut@index{#1\ix@style}#1}
```

Makro `\tit` vytiskne název přednášky.

```
\def\tit#1{\par\noindent#1\vspace{1ex}\par}
```

Od makra `\kwd` požadujeme pouze zapsání indexu do souboru klíčových slov. Jednotlivá klíčová slova jsou však oddělena čárkami. Proto za původní parametr přidáme čárku a tokeny `\kwd@\@nil` a zavoláme pomocné makro `\kwd@`. Toto makro má dva parametry. Prvním parametrem je text až do první čárky, druhým parametrem je text až do `\@nil`. Nejprve předáme první parametr makru `\kwd@index`. Pokud je druhým parametrem token `\kwd@`, pak již nemáme žádné další klíčové slovo. V opačném případě se `\kwd@` zavolá rekurzivně. Všimněte si, že rekurzivní volání je posledním příkazem rozvoje makra. Jinak by totiž záhy došlo k přeplnění paměti.

```
\def\kwd#1{\kwd@#1,\kwd@\@nil}
\def\kwd@#1,#2\@nil{
 \kwd@index{#1}
 \ifx#2\kwd@
  \let\next\relax
 \else
  \def\next{\kwd@#2\@nil}
 \fi\next}
```

Inicializace

Žádnou inicializaci v pravém smyslu slova vlastně nepotřebujeme. Pouze musíme nadefinovat názvy souborů.

```
\def\aut@file@nm{autindex}
\def\kwd@file@nm{kwdindex}
```

Definiční soubory pro *CsIndex*

Když jsme si nadefinovali indexová makra, musíme ještě dát příslušné instrukce programu ***CsIndex***. Autorský rejstřík je nutno zpracovávat jinak než rejstřík klíčových slov. Proto jsme v makrech `\aut@wrindex` a `\kwd@wrindex` použili odlišný „keyword“, takže nyní uživatel nemůže použít omylem nesprávný stylový soubor.

Pro zpracování autorského rejstříku použijeme následující definice:

```
  %% Toto je soubor ‘autind.ist’ pro demonstraci autorského
rejstříku
  %%
  keyword   "\\authorentry"
  preamble  "\n \\begin{theautindex} \n"
  postamble "\n\n \\end{theautindex}\n"
  item_x1   "\\efill \n \\subitem "
  item_x2   "\\efill \n \\subsubitem "
  delim_0   "\\pfill "
  delim_1   "\\pfill "
  delim_2   "\\pfill "
  heading_prefix   "{\\bf\\hfil "
  heading_suffix   "\\hfil}\\nopagebreak\n"
  headings_flag       1
  %%
  %% Konec souboru ‘autind.ist’.
```

Zpracování klíčových slov je velmi podobné. Stylový soubor se liší jen v detailech.

```
  %% Toto je soubor ‘kwdind.ist’ pro demonstraci
  %% rejstříku klíčových slov
  keyword   "\\keywordentry"
  preamble  "\n \\begin{thekwdindex} \n"
  postamble "\n\n \\end{thekwdindex}\n"
  item_x1   "\\efill \n \\subitem "
  item_x2   "\\efill \n \\subsubitem "
  delim_0   "\\pfill "
  delim_1   "\\pfill "
  delim_2   "\\pfill "
  heading_prefix   "{\\bf\\hfil "
  heading_suffix   "\\hfil}\\nopagebreak\n"
  headings_flag       1
  %%
  %% Konec souboru ‘kwdind.ist’.
```

Demonstrace sborníku

Teď si předvedeme, jak takový sborník vypadá, Je to skutečně jen krátká demonstrace s minimálním počtem přednášek. Delší text si jistě dovedete představit sami.

A Hodiny, jejich výzkum a využití

A.1 Markalous J. K., ∗Babočka A.
Studium toku písku v přesýpacích hodinách

A.2 ∗Kořízek M., Babočka A., Babočka B.
Nestacionární řešení Schrödingerovy rovnice pohybu kyvadla v nehomogenním gravitačním poli

B Hodiny, práce všeho druhu

B.3 ∗Pytlík B., Mravenec F.
Euklidovské metody opravy náramkových hodinek pomocí pravítka, kružítka a velké palice

B.4 Babočka A., ∗Babočka B.
Šetrné domlouvání zadřeným přesýpacím hodinám velkou palicí

Autorský rejstřík

Číslo vytištěné tučně odkazuje na přednášku, kde je příslušný autor hlavním autorem.

Klíčová slova

Jak zvládnout rozsáhlý rejstřík

Při vytváření velkého rejstříku se může stát, že si ***MakeIndex*** bude stěžovat na nedostatek paměti. Co se dá v takovém případě dělat? Zbývá nám pouze jediné východisko. Každý počítač má nějaký třídicí program. Můžeme tedy soubor `idx` předtřídit příslušným programem, výsledek rozdělit na menší části, ty zpracovat ***MakeIndex***em a výsledné soubory `ind` textovým editorem spojit a upravit. Bohužel, systémový třídicí program nemusí umět češtinu a dokonce se také může dostat do problémů s nedostatkem paměti. Pak vyžaduje předzpracování souboru `idx` větší objem ruční práce nebo si musíme vytvořit jiný pomocný program. Programátoři si jistě pomohou svým oblíbeným programovacím jazykem, ale tento problém lze naprogramovat i v TEXu. O tom si však povíme někdy příště.

Závěr

V tomto článku jsme se pokusili nastínit problematiku tvorby rejstříku v českém a slovenském jazyce a ukázali jsme možnost použití programů ***MakeIndex*** a ***C*****S*****Index*** v situacích, které nejsou popsány v dokumentaci. Je přirozené, že výklad zdaleka nebyl vyčerpávající. Možnosti aplikace těchto programů jsou omezeny pouze invencí uživatele.

Rejstřík maker

Čísla vytištěná italikou označují stránky, kde je příslušné makro popsáno. Podtržená čísla odkazují na definici a všechna ostatní ukazují místo, kde je makro použito.

Zdeněk Wagner
wagner@csearn

Ještě jednou \contparindent

Štěpán Kasal

Motto:
Jiná (a lepší) řešení jsou jistě možná.
Ladislav Lhotka

(Abych trochu kompenzoval drzost, jež čiší z motta, přiznám, že v době, kdy jsem článek četl, jsem téměř nebyl schopen to makro pochopit, natož abych takové obraty aktivně používal.)

V TEXbulletinu č. 1/92 byl velice zajímavý článek, který vysvětloval, jak lze zjistit skutečnou délku posledního řádku v odstavci. Jako příklad aplikace uváděl způsob sazby, ve kterém odsazení každého odstavce je rovno délce posledního řádku v předchozím odstavci.

Domnívám se však, že to nebyl vhodný příklad — `\contparindent` by měl sázet všechny odstavce (z hlediska lidského) do jednoho „odstavce“ (z hlediska TEXu). TEX totiž zpracovává celý odstavec najednou, protože polohy míst, ve kterých se přechází na nový řádek (*breakpoints*) jsou vzájemně závislé. V našem případě jsou ovšem takto provázány (téměř) všechny odstavce, proto je

nutno je sázet vcelku. Makro `\par` tedy předefinuji tak, aby vytvořilo volnou řádku, která může být téměř kdekoliv rozdělena.

Možná jste si v minulém odstavci všimli dvou slov „téměř“ a tážete se, co mají znamenat. To první souvisí s makrem `\contparindent` z článku Ládi Lhotky; snad bych ho měl připomenout:

```
1  \tolerance=800
2  \newdimen\oldParindent \oldParindent=\parindent
3  \def\cpar{\endgraf
4    \setbox1=\lastbox
5    \setbox2=\hbox{\unhcopy1}%
6    \global\dimen1=\baselineskip
7    \global\advance\dimen1 by -\dp2
8    \global\parindent=\ifdim .9\hsize>\wd2 \wd2
9      \else \oldParindent\fi
10   \nointerlineskip\box1\egroup\par
11   \unvbox0}
12 \def\contparindent{\bgroup\baselineskip=0pt
13   \everypar={\setbox0=\vbox\bgroup\normalbaselines
14      \let\par=\cpar
15     \everypar={}\indent\space
16     \vrule height\dimen1depth 0pt width 0pt}}
17 \def\normalparindent{\egroup\parindent=\oldParindent}
```

V řádcích 8 a 9 se v případě, že by první řádek dalšího odstavce měl být kratší než desetina `\hsize`, definuje `\parindent` jako `\oldParindent`. Něco takového se mi nepodařilo napsat dovnitř odstavce (nejspíš to není nijak jednoduše možné). Lze však samozřejmě lehko TEXu zakázat, aby dělil volnou řádku blíže k jejímu levému okraji než 0,1`\hsize`. (Když to způsobí `Overfull`, stačí v tom místě vložit `\normalparindent \contparindent`. Nyní konečně uvedu definice:

```
\tolerance=800
\newdimen\krok \krok=3pt
\def\blankline{\unskip \nobreak\hskip.1\hsize plus.5\krok
  \vadjust{\vskip\parskip}% we're in horizontal mode
  \dimen0=.9\hsize
  \loop  \penalty0 \vadjust{}\kern\krok
         \advance\dimen0 by -\krok
  \ifdim  \dimen0 > 0pt  \repeat
  \hbox{\hskip\dimen0 plus.5\krok}\ignorespaces }
\def\contparindent{\par\let\par\blankline}
\def\normalparindent{\let\par\endgraf\par}
```

Makro `\blankline` využívá v řádku 6 dvou fint: TEX nedělí v `\kern`u (ten může být totiž i uvnitř slova), pokud za ním není glue. Druhá finta spočívá v tom, že TEX za místem, kde rozdělil, ruší postupně všechno, co smí; jelikož však nesmí zrušit `\vadjust{}`, nezruší ani následující `\kern`y. Dále si musí hledět toho, aby v obou řádcích bylo dostatečně pružné glue. Hodnotě `\tolerance 800` přibližně odpovídá `plus 0,5\krok`, neboť v obou řádcích dohromady chybí právě `\krok`. (Když se glue musí roztáhnout o dvojnásobek hodnoty uvedené za `plus`, je to badness asi 800.)

Může ovšem lehko dojít k překročení paměťových možností, neboť TEX nutíme sázet jeden veliký odstavec. Já používám počítač 386 SX se 4 MB paměti a emTEX386 a podařilo se mi za 3 minuty takto vysadit 199 odstavců na 17 stránek formátu A5; 215 odstavců už jsem nevysadil. Lze však kamkoli dovnitř odstavce (jsme-li dost daleko od jeho začátku) vložit

```
{\parfillskip=0pt \par \parskip=0pt \noindent}.
```

V uvedené verzi ovšem každé `\par` vytvoří volný řádek — tedy máme-li ve zdrojovém textu dva volné řádky za sebou, objeví se volný řádek i ve výsledku; rovněž volný řádek před `\normalparindent` se projeví. Ve skutečnosti jsem používal tato makra:

```
\edef\originalcatcodeat{\the\catcode`\@}
\catcode`\@=11
\tolerance=800
\newdimen\krok \krok=3pt
\def\blankline{\unskip \nobreak\hskip.1\hsize minus.5\krok
  \vadjust{\vskip\parskip}%
  \dimen@=.9\hsize
  \loop  \penalty\z@ \vadjust{}\kern\krok
         \advance\dimen@ -\krok
  \ifdim  \dimen@ > \z@  \repeat
  \hbox{\hskip\dimen@ plus.5\krok}\ignorespaces
  \advance\count@\@ne \message{\the\count@.par}%
  }
\def\contparindent{\par\let\par\bl@nkline  \count@=\z@ }
\def\bl@nkline{\futurelet\next\bl@nklin@}
\def\bl@nklin@{\ifx\next\normalparindent \else
                  \ifx\next\par \else  \blankline \fi\fi}
\def\normalparindent{\let\par\endgraf\par}
\catcode`\@=\originalcatcodeat
```

Štěpán Kasal

Jak jsem se sázel svisle (viz podpis)

Oldřich Ulrych

Před několika týdny se ke mně obrátil kolega s ukázkami tabulek, které by bylo možné snadno vytvářet v TEXu, které však musí sázet pomocí jiného software pro DTP, neboť potřebuje mít v záhlaví tabulky svisle vysázené texty. Příklad tabulky, na které je demonstrován uvedený problém, je vysázen vpravo. Pokusy tisknout tabulku tak, že nejdříve vytiskneme svislé texty a pak vodorovné, selhávaly kvůli mechanickým vůlím papíru ve vodicí dráze tiskárny.

	Poledník	Rovnoběžka	Svačina	Oběd
Praha	15	25	10.30	12.20
Brno	18	24	10.45	13.00

Protože šlo o to, jak sázet v záhlaví tabulek pouze takový hladký text, ve kterém každý znak ze vstupního souboru je přímo znakem fontu (tj. není možné používat žádné složené objekty TEXu a tudíž ani ligatury), navrhl jsem celkem bezpracné řešení, které svislou sazbu umožňovalo. Toto řešení je založeno na následujících podmínkách a krocích:

1. Aby bylo možné sázet české texty, musíme použít osmibitové fonty, např. fonty DC Norberta Schwarze nebo CS fonty upravené Ladislavem Lhotkou a Karlem Horákem. Tato potřeba plyne z toho, že každý znak zdrojového textu musí být přímo znakem ve fontu.
2. Upravíme zdrojový text fontu tak, aby při generování METAFONTem se bitová reprezentace písmen vytvářela pootočená o 90° a generujeme font v požadovaném zvětšení.
3. Napsat makro pro TEX, které bude umisťovat písmena do sloupečku nad sebe tak, že zpracováním `dvi` souboru vznikne v požadovaných místech (s použitím fontu s pootočenými znaky) svislá sazba (přirozeně bez možnosti automatického dělení slov a zalamování do odstavců).

Podotkněme, že se nejedná o komplexní řešení vertikální sazby, ale je zde řešena pouze jedna velmi speciální úloha TEXem. Omezení, která z navrženého postupu vyplývají jsou natolik vážná, že je nutno tento příspěvek chápat spíše jako hru s METAFONTem a TEXem. Možných

postupů, jak sázet svisle existuje mnoho, tento se mi zdál pro daný účel nejjednodušší.

Jestliže jsme zhruba načrtli postup, věnujme se nyní všem detailům. Protože popis všech změn a úprav není dlouhý (včetně makra pro TEX) uveřejníme zde i všechny zdrojové texty. V dalším předpokládejme, že chceme sázet svislé texty fontem `dcr10` (desetibodová antikva z rodiny písem DC).

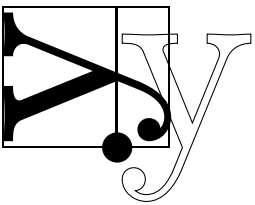
1. Zkopírujme soubor `dcr10.mf` např. do souboru `dcr10-v.mf` a na jeho začátek napišme příkaz

```
input vbase
```

2. V adresáři zdrojových textů pro METAFONT vytvořme zmíněný soubor `vbase.mf` s následujícím obsahem:

```
%  The changes in this file rotate characters in CM and DC fonts
%  90 degree counterclockwise.
%
extra_beginchar:="currenttransform:=identity slanted slant rotated 90;"&
              "def t_ = transformed currenttransform enddef;";
```

   Tyto řádky způsobí, že bitová mapa každého znaku, generovaného METAFONTem bude otočena o 90° proti směru hodinových ručiček. Toto je vidět na obrázku níž. Obrys písmene *y* znázorňuje výstup bez otočení, plné písmeno *y* je výsledkem výše uvedeného příkazu. Puntíkem je znázorněn referenční bod znaku (tento zůstává nezměněn, stejně jako výška, šířka a hloubka znaku i po otočení — otáčí se právě kolem referenčního bodu).
3. Generujeme font `dcr10-v.mf` pro požadované výstupní zařízení a požadované zvětšení.

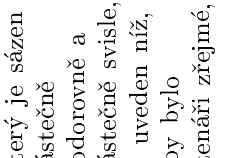


Tímto máme připraven font `dcr10-v`, kterým můžeme sázet svisle (zdola nahoru).

Celá idea vertikální sazby je založena na umisťování referenčních bodů jednotlivých znaků sázeného textu nad sebe s případným vložením mezery (kerningu). Navrhl jsem makro `downup.tex`, které obsahuje tyto definice:

`\downup{...}` je řídicí slovo s jedním argumentem, jímž je text, který se bude sázet zdola nahoru. Uvnitř tohoto textu je možné použít řídicí

znak \\ k naznačení zlomu řádku. V takovém případě jsou sázeny jednotlivé svislé řádky vedle sebe podle pravidel obvyklých při vkládání mezer mezi horizontální řádky ve vertikální seznamu (tj. využívají se hodnoty `\baselineskip`, `\lineskiplimit` a `\lineskip`).
`\fordownup{...}` je řídicí slovo s jedním argumentem, který se vkládá před začátek textu zpracovávaného řídicím slovem `\downup`.

Zdrojový text pro tento odstavec, 

jak jsme jej vysázeli:

```
\font\fsvisle=dcr10-v
\fordownup{\fsvisle}
Zdrojový text pro tento odstavec,
\downup{který je sázen\\ částečně \\vodorovně a\\ částečně svisle,}%
\downup{je uveden níž,\\ aby bylo \\ čtenáři zřejmé,} jak jsme jej
vysázeli:
```

Pro úplnost ještě uvádím celé makro `downup.tex` v podobě, v jaké zde bylo použito:

```
% Simple macro for vertical typesetting of pure text.
% For detail description, see the end of this file.
%
% Oldrich Ulrych                    October 9, 1992
% Mathematical Institute of Charles University
% e-mail: oulrych@cspguk11.bitnet
%
\edef\catcodeat{\the\catcode`\@ }     \catcode`\@=11
%
% definitions of new registers
%
\newbox\vertb@x             % box for vertical text
\newbox\auxb@x              % auxiliary box
\newdimen\m@xht             % "height" of vertical "row"
\newdimen\m@xdp             % "depth" of vertical "row"
\newdimen\@uxdim            % auxiliary dimension
\newdimen\pr@vd@pth \pr@vd@pth=0pt % "depth" of previous "row"
%
{\def\1{\global\let\spa@ce= } \1 }% \spa@ce is space
\def\\{}%                         % \\ is defined
```

```
\def\downup{\begingroup\f@rdownup % user's setting
    \ifvmode\leavevmode\fi        % to vboxes are set in line
    \m@xht=0pt                    % "left height" is zero
    \m@xdp=0pt                    % "right depth" is zero
    \setbox\vertb@x=\vbox{}%      % the box is empty initially
    \def\pr@vletter{}%            % previous letter is none
    \afterassignment\sp@cetreat   %            (for kerning)
    \let\next= }                  % eat the opening bracket
\def\sp@cetreat{\futurelet\next\t@stclosing} % next character
\def\t@stclosing{%
   \ifx\next\egroup
     \let\next\lastputvb@x  % all argument is scanned over
   \else
     \expandafter\ifx\next\spa@ce
       \let\next\@naspace   % next character is space
     \else
       \ifx\next\\%
         \let\next\separateb@x% next token is "line break"
       \else
         \let\next\@@tone   % other cases
   \fi\fi\fi\next}
\def\lastputvb@x{\putvb@x\endgroup % make last vbox
   \let\next= }             % and eat the closing bracket
\def\putvb@x{%              % make standard "height and depth" of box
   \@uxdim=-\pr@vd@pth
   \advance\@uxdim by -\m@xht
   \advance\@uxdim by \baselineskip
   \ifdim\@uxdim<\lineskiplimit
     \kern\lineskip
   \else
     \kern\@uxdim             % interline skip
   \fi
   \global\pr@vd@pth=\m@xdp
   \hbox{\kern\m@xht \box\vertb@x \kern\m@xdp}}% and form box properly
\def\@naspace{\setbox\vertb@x
  =\vbox{\kern1ex\unvbox\vertb@x}%  % leave vert space
     \def\pr@vletter{}%         % space doesn't influence the kerning
     \afterassignment\sp@cetreat\let\next= } % eat the space
\def\@@tone#1{\setbox\vertb@x=\vbox{\r@tletter{#1}\unvbox\vertb@x}%
   \def\pr@vletter{#1}\sp@cetreat}    % add the token to the vert. list
\def\r@tletter#1{\setbox\auxb@x=\hbox{\pr@vletter#1}%letters with kerning
   \@uxdim=-\wd\auxb@x                % remember their width
   \setbox\auxb@x=\hbox{\pr@vletter}% % take the previous letter
   \advance \@uxdim by \wd\auxb@x     % and subtract its width
   \setbox\auxb@x=\hbox{#1}%          % take the letter
```

```
   \advance \@uxdim by \wd\auxb@x     % and subtract its width
   \ifdim\m@xht<\ht\auxb@x \global\m@xht=\ht\auxb@x \fi % maximal height
   \ifdim\m@xdp<\dp\auxb@x \global\m@xdp=\dp\auxb@x \fi % maximal depth
   \ht\auxb@x=\wd\auxb@x \dp\auxb@x=0pt \wd\auxb@x=0pt% change its dim's
   \box\auxb@x                        % print letter
   \kern-\@uxdim}                     % make vertical kerning
\def\separateb@x#1{\putvb@x           % make vbox
   \def\pr@vletter{}%                 % prevletter is none
   \futurelet\next\trysp@ce}          % look ahead
\def\trysp@ce{\ifx\next\spa@ce
      \let\next\eat@neitem\else       % ignore following spaces
   \let\next\sp@cetreat\fi\next}
\def\eat@neitem{\afterassignment\sp@cetreat\let\next= } % eat space
\def\fordownup#1{\def\f@rdownup{#1}}
\fordownup{\tenrm}
%
\catcode`\@=\catcodeat        \let\catcodeat=\undefined
%
%%%%%%%%%%%%%%%%%%%%%%%%%%  C o m m e n t s  %%%%%%%%%%%%%%%%%%%%%%%%%%
%
%  \fordownup{...}  should contain the font for vertical printing
%  \downup{...}     is the simple text printed from bottom to the top
%                   (ligatures are printed as separated letters)
%  \\               can be used to separate lines in the argument
%                   of \downup
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Oldřich Ulrych

Elektronický archiv a další služby

Martin Bílý

Díky laskavému pochopení katedry počítačů ČVUT FEL je všem přátelům TEXu k dispozici několik typů služeb poskytovaných prostřednictvím počítačové sítě Internet. K dispozici je archiv přístupný pomocí služby **ftp** nebo prostřednictvím **mailserveru**. Dále je vedena **elektronická konference csTEX**.

Služby jsou poskytovány počítači typu VAX s operačním systémem VMS. Při práci s archivem je třeba dodržovat konvence VMS pro práci se soubory a adresáři. Ukažme si proto nejprve, o jaké konvence se jedná.

Operační systém VMS

S jistým zjednodušením platí, že specifikace souboru se v operačním systému VMS skládá z označení adresáře, jména souboru a typu souboru. Adresáře jsou hierarchicky uspořádány. Specifikace adresáře se uzavírá do hranatých závorek `[]`, přičemž jednotlivé podadresáře se navzájem oddělují tečkami. Máme-li na disku adresář `pub`, v němž je uložen podadresář `tex` obsahující další podadresář `fonts`, pak jeho specifikace je `[pub.tex.fonts]`. Použití malých nebo velkých písmen zde nebo v jiné části specifikace není významné.

I ve VMS se setkáváme s pojmem aktuální adresář. Předpokládejme, že v našem příkladu je aktuálním adresářem `[pub.tex]`. Jeho podadresář `fonts` pak můžeme snadno specifikovat jako `[.fonts]`. Uvedení tečky před jménem adresáře napovídá, že se jedná o specifikaci vůči aktuálnímu adresáři. Na nadřízený adresář se lze odkázat pomocí `[-]`. Pokud je třeba odkázat se na adresář o několik úrovní výše, stačí uvést příslušný počet znamének minus. Již uvedené možnosti lze dále kombinovat. Předpokládejme, že v adresáři `pub` je vedle adresáře `tex` ještě podadresář `standards`, který dále obsahuje adresář `rfc`. Adresář `rfc` vůči našemu aktuálnímu adresáři můžeme specifikovat jako `[-.standards.rfc]`.

Jméno a typ souboru se navzájem oddělují tečkou. Soubory typu `.dir` nejsou běžné soubory, ale podadresáře. Při výpisu adresáře lze používat i nejednoznačnou specifikaci souboru s využitím znaku hvězdička. Za zmínku snad stojí, že je povoleno třeba i `*tt*.mf`. V tomto příkladu se jedná o specifikaci všech souborů typu `mf`, v jejichž jménu se vyskytuje posloupnost dvou písmen `t`.

Ftp server

Archiv je přístupný pomocí služby **ftp** na adrese `vax.felk.cvut.cs`. Pro získání přístupu je třeba přihlásit se jako uživatel `anonymous` a uvést heslo `anonymous`.

Oproti jiným ftp serverům zde platí, že příkaz `dir` vypisuje několikařádkové údaje o každém souboru, takže pro běžného uživatele není vhodný. Příkaz `ls` vypisuje jen samotná jména souborů.

TEXový materiál je jen jednou z částí archivu. Je uložen v adresáři `[pub.tex]` a jeho podadresářích. Přírůstky za poslední období se evidují v souborech `00last7days.files` a `00last30days.files`. Pokud byste sami chtěli do archivu něčím přispět, je k dispozici adresář `[pub.income]`, kam svůj dar můžete vložit. Připojte prosím i několik vysvětlujících řádek, které můžete také poslat na moji adresu nebo na adresu správce všeobecné části archivu, jímž je Jan Schmidt, `schmidt@cs.felk.cvut.cs`.

Mailserver

Na adrese `mailserv@vax.felk.cvut.cs` je poskytována služba zvaná mailserver. Všechny dopisy posílané na tuto adresu se neukládají do poštovní schránky uživatele `mailserv`, ale předkládají se ke zpracování zvláštnímu programu, který se obvykle nazývá mailserver. Jeho úkolem je číst jednotlivé řádky přijatého dopisu a provádět akce, které jsou na nich předepsány. Na každém řádku může být zapsán jeden příkaz s případnými parametry. Server rozumí následujícím příkazům:

`help` nebo `?`
Po zadání příkazu mailserver zašle nápovědu, jak jej lze používat. Nápověda obsahuje popis jednotlivých příkazů a i příklady jejich použití.

`cd`
Příkazem pro změnu aktuálního adresáře se lze volně pohybovat po celém adresářovém stromu archivu. Jeho smyslem je zjednodušit příkazy dir a send tak, aby v nich nebylo nutné stále uvádět specifikaci adresáře.

`dir` nebo `ls`
Výpis obsahu adresáře. Jako nepovinný parametr lze uvést specifikaci adresáře, kterou je možno rozšířit o specifikaci souboru, obvykle nejednoznačnou. V odpovědi je uvedena specifikace souboru, jeho velikost v násobcích 512 bytů a datum vzniku souboru.

`ascii` nebo `text`
Nastavení režimu přenosu pro textové soubory. Tento režim je nastaven před zpracováním vašeho prvního příkazu.

`binary`
Nastavení pro přenos binárních souborů. Binární soubory jsou před odesláním kódovány do textového tvaru programem uuencode.

`get` *filename* nebo *send filename*
Požadavek na zaslání souboru, jehož specifikace je *filename*. Může se jednat o nejednoznačnou specifikaci. Rozsáhlé soubory se posílají rozdělené do několika samostatných dopisů. Binární soubory jsou před odesláním kódovány pomocí uuencode.

Pokud vám na terminálech IBM činí problémy psát hranaté závorky, v případě mailserveru místo nich lze používat znaménka menší, větší `<>`.

V další verzi mailserveru bude k dispozici zvláštní příkaz pro zpřístupnění neveřejné části archivu, ve které bude materiál šířený pouze členům CSTUGu.

Elektronická konference

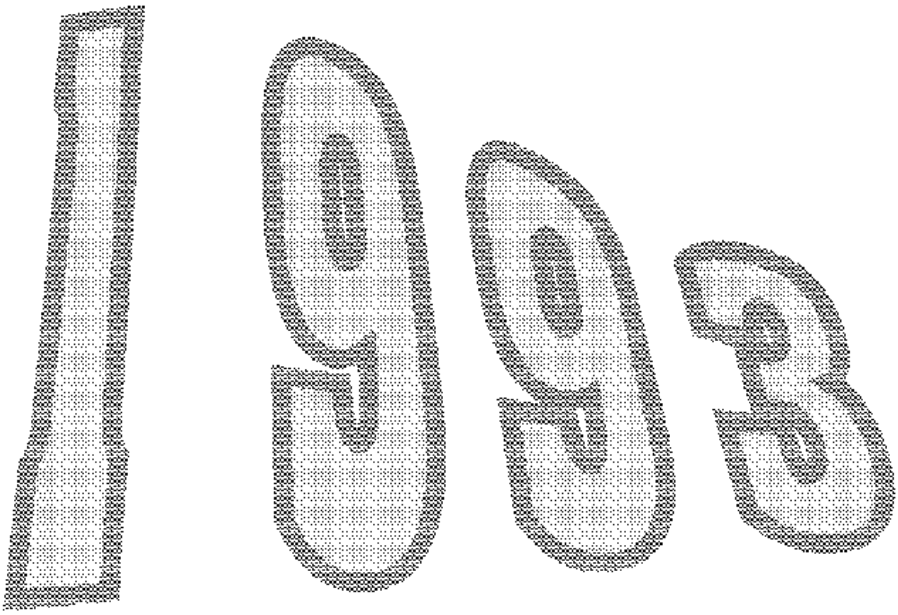
Na počítači `cs.felk.cvut.cs` je vedena elektronická konference věnovaná TEXu. Její jméno je `csTeX`. Konference je vedena v českém a slovenském jazyce. Jejím posláním je usnadnit náš vzájemný styk v diskusích o používání TEXu v našich národních podmínkách a s využitím nám dostupné techniky.

Do konference se lze přihlásit dvěma způsoby. První z nich používá konvence zavedené v síti Bitnet: Na adresu `listserv@cs.felk.cvut.cs` poslat mail obsahující příkaz `subscribe csTeX`. V Internetové variantě lze na adresu `csTeX-request@cs.felk.cvut.cs` poslat pouze příkaz `subscribe`. Po přihlášení uživatel obdrží uvítací dopis a seznam použitelných příkazů, které může na výše uvedené administrativní adresy zasílat. Příspěvky do konference se posílají na adresu `csTex@cs.felk.cvut.cs`, odkud jsou distribuovány všem účastníkům konference.

Martin Bílý
`bily@cs.felk.cvut.cs`

Z obsahu příštího čísla

Joachim Schrod: Komponenty TEXu
Jiří Veselý: Tabulky v plainTEXu
Zdeněk Wagner: Nerovnoběžné rovnoběžky
Jiří Rybička: Automatizovaná tvorba formulářů pomocí LATEXu



Vydalo:	Československé sdružení uživatelů TEXu vlastním nákladem jako interní publikaci
Počet výtisků:	600
Tisk:	HBT-Jet PRESS Neratovice
Adresa:	CSTUG MÚ UK, Sokolovská 83, 186 00 Praha 8