Zpravodaj Československého sdružení uživatelů TeXu

Zdeněk Wagner

Tvorba rejstříku

Zpravodaj Československého sdružení uživatelů TeXu, Vol. 2 (1992), No. 4, 176–198

Persistent URL: http://dml.cz/dmlcz/149645

```
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