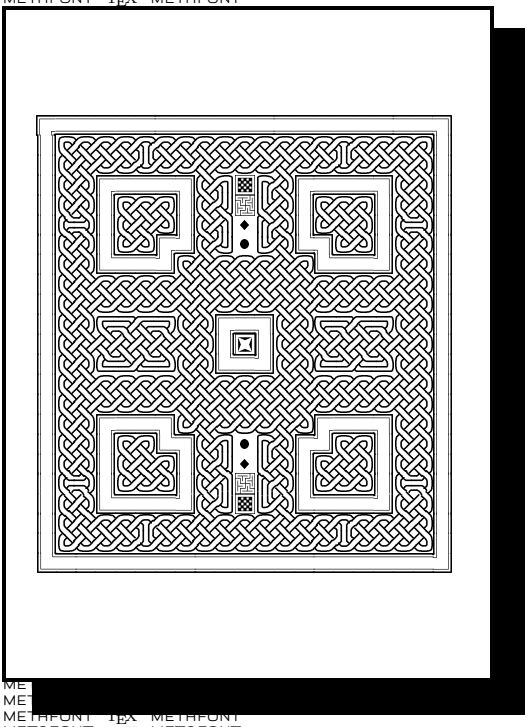
TEX METAFONT



3

ZPRAVODAJ 97

ČESKOSLOVENSKÉHO SDRUŽENÍ UŽIVATELŮ TEXU

ISSN 1211-6661 Ročník 7

OBSAH

Zpravodaj Československého sdružení uživatelů TEXu je vydáván v tištěné podobě a distribuován zdarma členům sdružení. Po uplynutí dvanácti měsíců od tištěného vydání je poskytován v elektronické podobě (PDF) ve veřejně přístupném archívu dostupném přes `http://www.cstug.cz`.

Své příspěvky do Zpravodaje můžete zasílat v elektronické podobě anonymním ftp na `ftp.icpf.cas.cz` do adresáře `/wagner/incoming/`, nejlépe jako jeden archivní soubor (`.zip, .arj, .tar.gz`). Současně zašlete elektronickou poštou upozornění na `mailto:bulletin@cstug.cz`. Uvedený adresář je pro vás „write/only“. Pokud nemáte přístup na Internet, můžete zaslat příspěvek na disketě na adresu:

Zdeněk Wagner
Vinohradská 114
130 00 Praha 3

Disketa musí být formátována pro DOS. Nezapomeňte přiložit všechny soubory, které dokument načítá (s výjimkou standardních součástí CSTEXu), zejména v případě, kdy vás nelze kontaktovat e-mailem.

ISSN 1211-6661

EncTEX — změny konverzních tabulek TEXu

Petr Olšák

Motto: Certainly, if I were a publishing house, if I were in the publishing bussiness myself, I would have probably had ten different versions of TEX by now for ten different complicated projects that had come in. They would all look almost the same as TEX, but no one else would have this program—they wouldn't need it, they're not doing exactly the book that my publishing house was doing.

Donald E. Knuth, Prague, March 1996

Článek popisuje jednoduchou úpravu TEXu, která umožňuje přímý přístup k interním vektorům `xord` a `xchr` používaným pro konverzi mezi vstupním kódováním a vnitřním kódováním TEXu. Například uživatelé emTEXu v DOSu a OS/2 mohou zasahovat do těchto vektorů prostřednictvím tzv. `tcp` tabulek, zatímco v UNIXu takto pružné řešení implicitně neexistovalo. Popsaná modifikace TEXu víceméně zastupuje vlastnosti TCP tabulek. Protože je tato modifikace provedena na úrovni změnového souboru `tex.ch` v jazyce WEB, je použitelná všude tam, kde se TEX implementuje ze zdrojových textů. Odzkoušena byla v UNIXových systémech s implementací TEXu `web2c`.

Na těchto implementacích TEXu jsme dosud měli dvě možnosti, jak ovlivnit kódovací algoritmy TEXu. První možností bylo použití záplaty pana Škarvady [3]. Toto řešení mělo přímo v sobě zabudováno několik kódových tabulek, ale tyto tabulky nebylo snadné měnit uživatelem, protože byly součástí zdrojového kódu k TEXu. Načtené kódování se neukládalo po inicializaci do formátu. To mi nepřipadalo příliš pružné.

Druhá možnost spočívala v použití tzv. `tcx` souborů, které byly obdobou známých `tcp` tabulek. Tato možnost ale byla ve zdrojovém kódu TEXu odkomentovaná se slovy: „`tcx` files are probably a bad idea, since they make TEX source documents unportable. Try the `inputenc` LATEX package.“ S těmito slovy bych byl ochoten polemizovat. Myslím si, že kdyby uživatelé UNIXových instalací našli obdobu `tcp` tabulek, pak by to zvláště v našich zeměpisných šířkách mnozí uvítali. Autor zmíněného sdělení o „špatné ideji“ asi nepoužívá český jazyk. Kdyby jej používal, kdyby byl nucen číst `logy` plné nesrozumitelných střížek nebo zapsané v odlišném kódování a kdyby nemohl psát kontrolní sekvence ve svém jazyce (například `\příkaz`), asi by si znovu rozmyslel, jakou funkci přidělil vektorům `xord` a `xchr` sám autor TEXu. Možná by pak přestal hovořit o „špatné ideji“. Já osobně mám z citovaného sdělení pocit, že se automaticky předpokládá, že TEX je používán vždy jen prostřednictvím LATEXu. Myslím si, že právě to je „špatná idea“.

Bohužel, dokumentace k `tcx` souborům zcela chybí, takže jen letmým pohledem do zdrojového kódu lze tušit, že to také ukládalo kódovací vektory čtené z `tcx` souborů do formátu a tím bylo možné instalovat formáty závislé na kódování, například `csplain.fmt` a `csplain-kam.fmt`.

Moje úprava zdrojového kódu `tex.ch` v instalaci `web2c` jde ještě dál, protože zahrnuje výhody druhého přístupu, a navíc jsou veškeré tabulky načítány za běhu TEXu jednoduše prostřednictvím `\input`. Implementoval jsem totiž do TEXu tři nové primitivy `\xordcode`, `\xchrcode` a `\xprncode`, pomocí nichž je možný přímý přístup k vektorům `xord` a `xchr`. Novinkou je též skutečnost, že se stávají tyto vektory zcela nezávislými, což otevírá další široké možnosti, které v předchozích řešeních nebyly vůbec možné.

Technický úvod do problematiky

Vektory `xord` a `xchr` mají velikost 256 bytů a obsahují informaci o překódování znaku vstupujícího do TEXu nebo vystupujícího na terminál a do textových souborů. Jedná se o pole vestavěná do programu, přes která jsou filtrovány veškeré textové vstupní a výstupní informace. Má-li znak na vstupu kód x a chceme, aby měl uvnitř TEXu kód y, pak musí být nastaven vektor `xord` tak, aby `xord[`x`]` $= y$. Při zpětném výstupu znaku na terminál a do logu a do souborů zpracovávaných pomocí `\write` platí tato pravidla: Není-li znak s kódem y označen jako „tisknutelný“, pak vystupuje pomocí přepisu `^^`*kód* y. Je-li tisknutelný, pak vystupuje s kódem $x =$ `xchr[`y`]`.

Standardně bývají v systémech s kódem ASCII nastaveny hodnoty těchto vektorů tak, že `xord[`i`]` $=$ `xchr[`i`]` $= i$ pro všechna $i \in 0 \ldots 255$. Na systémech, které nepoužívají ASCII, se může mapovat 94 tisknutelných ASCII znaků jinam. Mimoto je deklarována vlastnost „tisknutelnosti“ znaku v ASCII takto: Znak je tisknutelný, pokud má kód $y \in 32 \ldots 126$. Ostatní znaky se považují za netisknutelné a TEX je standardně přepisuje pomocí dvojité stříšky.

Instalace balíčku encTEX byla odzkoušena na distribuci TEXu `web2c` verze 7. Máme-li k této distribuci zdrojové texty, pak stačí spustit `patch <enctex.patch` v adresáři se souborem `tex.ch` a překompilovat TEX pomocí `make tex`. Instalace je podrobněji popsána v souboru `INSTALL`.

Aplikací přiložené záplaty `patch` se pozmění jediný soubor `tex.ch`. Ostatní soubory včetně pomocných knihoven v jazyce C zůstávají beze změny. Při činnosti „`make tex`“ se provede `tangle` na hlavní zdrojový soubor `tex.web` a pozměněný změnový soubor `tex.ch`. Tím vzniká Pascalský kód TEXu `tex.p`, který je pak konvertován do jazyka C skriptem `convert` a kompilován do spustitelné podoby.

Protože je veškerá změna TEXu implementována pouze do změnového souboru `tex.ch`, je vlastně toto řešení nezávislé na použitém operačním systému či

implementaci TEXu. Na jiných implementacích je ale jiný výchozí soubor `tex.ch`, takže zde nelze jednoduše provést povel `patch`. Proto je v balíčku zahrnut soubor `enctex.ch`, který obsahuje ty části změnového souboru `tex.ch`, které se týkají encTEXu. Veškeré změny vycházejí ze standardního zdrojového kódu TEXu `tex.web` až na jedinou výjimku: ukládání vektorů `xord` a `xchr` do formátu a čtení z formátu. Zde jsem s výhodou použil již hotové pomocné funkce *dump_things* a *undump_things* napsané pro `web2c` přímo v jazyce C. Pokud by to někdo potřeboval v jiné implementaci, musel by zřejmě použít analogii k *dump_four_ASCII* z `tex.web`.

Po instalaci balíčku je možno přímo nastavovat a číst obsahy vektorů `xord` a `xchr` prostřednictvím primitivů `\xordcode` a `\xchrcode` a dále nastavovat vlastnost „tisknutelnosti“ znaku pomocí primitivu `\xprncode`. Syntaxe všech tří nových primitivů je naprosto stejná, jakou známe například u primitivů `\lccode` a `\uccode`. Například:

```
\xordcode"AB="CD  \xchrcode\xordcode"AB="AB \the\xchrcode200
```

nastavuje `xord[0xAB]=0xCD; xchr[xord[0xAB]]=0xAB` a dále vytiskne hodnotu `xchr[200]`.

Na rozdíl od podobných primitivů `\catcode`, `\lccode`, `\sfcode` a dalších však nově zavedené primitivy mají jednu podstatnou výjimku. Reprezentují interní registry TEXu, které vždy mají globální platnost. Proto je nastavení `\xordcode` a `\xchrcode` uvnitř skupiny za všech okolností globální, ačkoli to explicitně nepíšeme. Ústupem z požadavku na možnost lokálního deklarování hodnot jsem dosáhl podstatně větší efektivity výsledného kódu programu.

Primitiv `\xprncode` umožňuje nastavovat vlastnost „tisknutelnosti“ znaku takto: Znak s kódem y je tisknutelný právě tehdy, když je $y \in \{32 \ldots 126\}$ nebo `\xprncode`$y > 0$. Napíšeme-li například `\xprncode255=1`, bude tisknutelný znak s kódem 255. Na druhé straně, nastavení `\xprncode‘a` třeba na nulu nemá na chování programu žádný vliv, protože kód znaku „`a`“ je v množině $\{32 \ldots 126\}$. Tímto opatřením program vykazuje určitý pud sebezáchovy, protože zlý uživatel by mu mohl nastavit všechny znaky jako netisknutelné a program by ztratil schopnost se vyjadřovat. Hodnoty `\xprncode` lze nastavit jako u ostatních nových primitivů v rozsahu nula až 255, ovšem otázka tisknutelnosti je totožná s otázkou na kladnou hodnotu bez ohledu na to, jak velká tato hodnota je.

Výchozí hodnoty pro kódování v době iniTEXu jsou následující:

`\xordcode`$\,i = i$ pro $i \in \{128 \ldots 255\}$,
`\xchrcode`$\,i = i$ pro $i \in \{128 \ldots 255\}$,
`\xprncode`$\,i = 0$ pro $i \in \{0 \ldots 31, 127 \ldots 255\}$,
`\xprncode`$\,i = 1$ pro $i \in \{32 \ldots 126\}$.

Hodnoty `\xordcode`$\,i$ a `\xchrcode`$\,i$ pro $i \in \{0 \ldots 127\}$ jsou závislé na operačním systému. Pokud systém pracuje v rámci této množiny s ASCII kódováním

(což je obvyklé), pak je `\xordcode` $i = i$ a `\xchrcode` $i = i$. Na jiných systémech jde hlavně o to, jak systém kóduje 95 základních tisknutelných znaků, které jsou v ASCII na pozicích $\{32 \ldots 126\}$.

Hodnoty `\xordcode`, `\xchrcode` a `\xprncode` se v této úpravě TEXu ukládají do formátu `fmt`, z něhož jsou znovu načteny při běhu produkční verze TEXu.

Než budeme zasahovat do hodnot vektorů `xord` a `xchr`, je dobré si uvědomit několik souvislostí. K tomu použiji matematické značení a terminologii. Nechť $X = \{0 \ldots 255\}$ je množina obsahující vstupní kódy a $Y = X$ je matematicky tatáž množina, ovšem budeme ji používat pro označení vnitřních kódů TEXu. Nechť dále $Y_p \subseteq Y$ je množina všech tisknutelných znaků. Je:

$$Y_p = \{y; \texttt{\textbackslash xprncode}\, y > 0\} \cup \{32 \ldots 126\}.$$

Je zřejmé, že nastavení hodnot vektoru `xchr` na množině $Y \setminus Y_p$ nemá pro chování programu žádný vliv. Dále označme vstupní zobrazení $I : X \to Y$ definované hodnotami vektoru `xord` a výstupní zobrazení $O : Y_p \to X$ definované hodnotami vektoru `xchr`. Výchozí nastavení je voleno tak, že I je prosté a na a O je prosté. Nastavíme-li `\xprncode` tak, že $Y_p = Y$, pak i zobrazení O je prosté a na. Navíc je inverzní k zobrazení I. Ovšem tento ideál platí pouze do prvního zásahu do vektorů `xord` a `xchr`. Nechť například máme $x \neq y$ a $x \in X$, $y \in Y$. Proveďme jedno elementární přehození:

$$\texttt{xord}[x] = y, \quad \texttt{xchr}[y] = x. \tag{1}$$

Nyní není ani zobrazení I ani O prosté. Je totiž `xord[`x`]` = `xord[`y`]` a totéž platí pro vektor `xchr`. Aby byla po dalších n elementárních přehozeních naše zobrazení opět prostá, musí existovat posloupnost $x_0, \ldots x_n$ taková, že

$$x_0 = x, \quad x_1 = y \quad \text{a} \quad \texttt{xord}[x_i] = x_{i+1} \quad \text{pro všechna } i \in \{0 \ldots n-1\}$$

a dále musí `xord[`x_n`]` $= x$. Podobná vlastnost musí platit pro vektor `xchr`. Často se při prohazování kódů zaměřujeme jen na podmnožinu X (například abecední znaky, tisknutelné znaky, dosažitelné znaky v daném kódování). Pak se nesmíme divit, že výsledek našeho nastavení není prosté zobrazení a tím není splněna ani podmínka, že $O = I^{-1}$, přestože jsme důsledně prováděli elementární přehození podle (1) pro oba vektory vždy současně. Většinou nám ale skutečnost, že nejsou naše zobrazení prostá, v dané aplikaci moc nevadí.

Kódovací tabulky

Protože změna vektorů `xord` a `xchr` může totálně rozhodit chování TEXu zcela k nepoznání, doporučuji používat určité soubory, které nastaví požadované kódování, a dále s primitivy `\xordcode`, `\xchrcode` a `\xprncode` za běhu TEXu

moc nelaškovat. V balíčku encTEX jsou k dispozici soubory, které změnu vektorů pro běžná kódování definují. Tyto soubory mají obvyklou příponu `tex`. Říkáme jim kódovací tabulky. Rozlišujeme dva typy kódovacích tabulek.

První typ kódovacích tabulek

První typ tabulek deklaruje vnitřní kódování TEXu ve vztahu ke kódování, které je běžně používané v hostitelském operačním systému. Máme-li například v systému kódování ISO-8859-2 a vnitřní kódování TEXu volíme podle Corku (kódování je označováno jako T1), pak tabulka musí předefinovat `xord` vektor tak, aby mapoval znaky z ISO-8859-2 do T1 a vektor `xchr` musí převádět zpátky z T1 do kódování systému.

Tento typ tabulek obsahuje v názvu souboru vstupní i cílové vnitřní kódování TEXu. Například tabulka `iso2-t1.tex` vypadá následovně:

```
%%% The encoding table, v.Aug.1997 (C) Petr Ol\v s\'ak
%%% input: ISO-8859-2, internal TeX: T1 alias Cork

\input encmacro  \input t1macro

%            input  TeX   lc   uc  cat       sequence
\setcharcode   "C1  "C1  "E1  "C1  11  \texaccent \'A
\setcharcode   "E1  "E1  "E1  "C1  11  \texaccent \'a
\setcharcode   "C4  "C4  "E4  "C4  11  \texaccent \"A
\setcharcode   "E4  "E4  "E4  "C4  11  \texaccent \"a
\setcharcode   "C8  "83  "A3  "83  11  \texaccent \v C
\setcharcode   "E8  "A3  "A3  "83  11  \texaccent \v c
\setcharcode   "CF  "84  "A4  "84  11  \texaccent \v D
...
\setcharcode   "A7  "9F    0    0  12  \texmacro  \S
\setactivecode "D7  "03   {$\times$}
...
\redefaccent \'
\redefaccent \v
...
```

Každá tabulka prvního typu čte soubor `encmacro` s definicemi maker `\setcharcode`, `\setactivecode`, `\texaccent`, `\texmacro` a `\redefaccent`.

- `\setcharcode #1 #2 #3 #4 #5` deklaruje TEXovské kódy pro jeden znak. Nastaví `xord[#1]=#2`, `xchr[#2]=#1`, `\xprncode#2=1` a postupně nastaví `\lccode`, `\uccode` a `\catcode` znaku s kódem `#2` na hodnoty `#3`, `#4` a `#5`. Je-li `#1` otazník, pak se `xord` a `xchr` nenastaví a `\xprncode=0`.

• `\setactivecode #1 #2 {`*definice*`}` nastaví rovněž hodnoty kódovacích vektorů `xord[#1]=#2`, `xchr[#2]=#1`, `\xprncode#2=1` a dále znak s kódem `#2` deklaruje jako aktivní a definuje jej podle *definice*. Je-li `#1` otazník, `\setactivecode` neprovede vůbec nic. Například v ISO-8859-2 je na pozici `"D7` znak „×“. My jsme v naší ukázce tomuto znaku přidělili vnitřní kód TEXu `"03` a znak bude expandovat na `$\times$`.

• `\texaccent #1` připravuje expanzi zápisu `#1` na znak s kódem `#2` z naposledy použitého `\setcharcode`. Například zápis `\v C` bude podle naší ukázky expandovat na znak s kódem `"83`. Pokud zápis pro akcent není v tabulce uveden, zůstává v původním významu, tj. třeba `\v g` expanduje na primitiv `\accent`, který usadí háček nad písmeno g. K aktivaci všech zápisů `\texaccent` dojde až po použití makra `\redefaccent` (viz níže).

• `\texmacro #1` deklaruje makro `#1` tak, že bude expandovat na znak s kódem `#2` z naposledy použitého `\setcharcode`. K předefinování makra `#1` dojde (na rozdíl od `\texaccent`) okamžitě. Například makro `\S` bude podle naší ukázky expandovat na znak s kódem `"9F`, protože na této pozici je podle Corku znak „§“.

• `\redefaccent #1` aktivuje expanzi zápisů podle `\texaccent` pro jeden konkrétní akcent `#1`.

Kromě toho je na začátku tabulky čten soubor definic závislých na kódování textového fontu TEXu. V naší ukázce jde například o soubor `t1macro`. Definují se tam sekvence `\promile`, `\clqq` a další. Jedná se o jednoduchou obdobu `fd` souborů z NFSS.

Může se stát, že nechceme uvedená makra použít, ale hodnoty z tabulky načíst chceme. Pak můžeme přistoupit k následujícímu triku: Definujeme si makra `\setcharcode` až `\redefaccent` sami a dále provedeme načtení tabulky takto:

```
\let\originput=\input \def\input #1 {}
\originput il2-t1
\let\input=\originput
```

V balíčku encTEX jsou připraveny tyto tabulky prvního druhu:

Název souboru	vstupní kódování	vnitřní kódování TEXu
`il2-csf.tex`	ISO8859-2	$\mathcal{CS}$-font
`kam-csf.tex`	Kamenických	$\mathcal{CS}$-font
`1250-csf.tex`	CP1250, MS-Windows	$\mathcal{CS}$-font
`852-csf.tex`	CP852, PC Latin2	$\mathcal{CS}$-font
`il2-t1.tex`	ISO8859-2	T1 alias Cork
`kam-t1.tex`	Kamenických	T1 alias Cork
`1250-t1.tex`	CP1250, MS-Windows	T1 alias Cork
`852-t1.tex`	CP852, PC Latin2	T1 alias Cork

Za zmínku stojí první uvedená tabulka `il2-csf.tex`, protože ta jediná ponechává vektory `xord` a `xchr` beze změny. Tuto tabulku je tedy možné použít i v TEXu, který neobsahuje rozšíření encTEX. Pro formát `csplain` by příslušný soubor `csplain.ini` mohl vypadat třeba takto:

```
\input csfonts    % re-defines primitive \font
\input plain      % format Plain
\restorefont      % original meaning of primitive \font
\input il2-csf    % input ISO8859-2, internal TeX: CS-font
\input hyphen.lan % czech / slovak hyphenation pattern
\input plaina4    % \hsize and \vsize for A4
\everyjob=\expandafter{\the\everyjob
  \message{The format: plain-il2-cs <Sep. 1997>.}
  \message{The cs-fonts are preloaded and A4 size predefined.}}
\dump
```

Takto generovaný formát by se od standardního formátu `csplain` neměl mnoho lišit. Pokud v tomto `ini` souboru zaměníme kupříkladu slovo `il2` za `1250`, pak vytvoříme identický formát, který je ale namísto UNIXu určen pro prostředí MS-Windows. V tomto případě ovšem už budeme potřebovat rozšíření TEXu s názvem encTEX.

Pokud bychom chtěli vytvořit analogii formátu `csplain` pro použití s DC/EC fonty, pak bych doporučoval tento postup:

```
\input noprefnt   % \font\preloaded is not preloaded
\input plain      % format Plain
\restorefont      % original meaning of primitive \font
\input dcfonts    % load text-style fonts
\input il2-t1     % input ISO8859-2, internal TeX: Cork
\input hyphen.lan % czech / slovak hyphenation pattern
\input plaina4    % \hsize and \vsize for A4
\everyjob=\expandafter{\the\everyjob
   \message{The format: plain-il2-dc <Sep. 1997>.}
   \message{The dc+cm-fonts are preloaded and A4 size predefined.}}
\dump
```

Necháme tedy matematické fonty načíst originálním makrem `plain` a potom předefinujeme textové fonty v souboru `dcfonts` jednoduše takto:

```
\font\tenrm=dcr10  \font\tenbf=dcbx10  \font\tenit=dcti10
\font\tentt=dctt10 \font\tensl=dcsl10  \tenrm
```

Upozorňuji, že tento formát může v sobě skrývat podstatné odlišnosti od originálního formátu `csplain`. Buďte při jeho používání opatrní.

Pokud budeme chtít vytvořit další tabulky, které mají stejné vnitřní kódování ale odlišné vstupní, pak většinou stačí vyjít z existující tabulky a pozměnit jen první sloupec.

Druhý typ kódovacích tabulek

Druhý typ tabulek provádí překódování pouze na vstupní straně TEXu. Poznáme je podle toho, že nemají na konci názvu značku pro vnitřní kódování TEXu (tj. `t1` nebo `csf`), ale značku používanou pro kódování operačního systému (např. `il2`, `kam`). Třeba tabulka `kam-il2.tex` provádí na vstupní straně konverzi z kódování kamenických do kódování ISO8859-2. Tento typ tabulek pozměňuje pouze vektor `xchr`, ale výstupní vektor `xord` ponechává beze změny. Takovou tabulku použijeme, pokud TEXem načítáme soubor, který je v jiném kódování, než běžně používáme na našem operačním systému. Přitom výstup do `log`, `aux` apod. ponecháme v kódování podle našeho systému. Tyto změny kódování je možné provádět i v průběhu zpracování jediného dokumentu.

Druhý typ tabulek navazuje na vstupní kódování deklarované dříve tabulkou prvního typu. Nastavení vnitřního kódování TEXu není vůbec druhým typem tabulek měněno. Uvedeme příklad. Při generování formátu jsme použili tabulku prvního typu `il2-t1.tex`, takže vnitřní kódování máme podle Corku. Nyní můžeme při zpracování dokumentu na přechodnou dobu vybrat některou z tabulek `*-il2.tex`, třeba:

```
\input kam-il2
\input dokument
\restoreinputencoding
nyní mohu pracovat v~původním kódování...
```

V době, kdy probíhá načítání souboru `dokument.tex` se provádí překódování z Kamenických do T1, uvnitř TEXu se vše zpracovává v T1 a výstup na terminál a do `log`u máme v ISO8859-2. V tomto kódování je také zapsán další text pod `\restoreinputencoding`. Tabulka totiž deklaruje toto makro, aby byl možný návrat k původnímu nastavení vektoru `xord`.

Při použití tabulek druhého typu musíme dát velký pozor, abychom něco neudělali špatně. V našem příkladě jsou všechny výstupy do souborů typu `aux` v ISO-8859-2, takže je při opakovaném spuštění TEXu nesmíme načítat v okamžiku, kdy máme nastaven vstupní kód podle Kamenických. To je také důvod, proč nedoporučuji generovat formát příkazem `\dump` v situaci, kdy máme načtenou tabulku druhého typu.

O kompatibilitě neboli slučitelnosti

Upravený TEX (encTEX) projde zcela bez problémů testem TRIP s výjimkou jediného případu: počet vložených „`multiletter control sequences`“ je o tři větší, než v originálním TEXu.

Veškeré změny TEXu, které nemění dosavadní chování TEXu, ale pouze přidávají nové primitivy, jsou zpětně zcela kompatibilní s originálním TEXem. Tím se myslí, že dokumenty napsané pro originální TEX se budou v pozměněném TEXu chovat zcela stejně. Výjimkou je snad dokument s konstrukcí typu `\ifx\xordcode\undefined`, ale četnost výskytu takových konstrukcí v dokumentech pro originální TEX je prakticky nulová.

Je třeba si ale uvědomit, že v pozměněném TEXu lze psát dokumenty a makra, které nejsou s originálním TEXem kompatibilní. To je nevýhoda všech rozšíření, která přidávají primitivy. Je zcela jedno, zda jsou nové primitivy přidány do TEXu „natvrdo“ (jako v případě encTEXu nebo pdfTEXu), nebo zda je přístup k novým primitivům otevřen až po speciální volbě na příkazovém řádku při inicializaci formátu (NTS, MLTEX). Zde záleží na tom, jak moc se používání nových primitivů rozšíří, a kdo bude dohlížet, aby rozšířená množina primitivů byla celosvětově standardizována. V případě mého jednoduchého rozšíření si nekladu žádný nárok na to, aby se to dostalo do nějakých standardů. Prostě jsem si to udělal pro svoje potřeby a pokud se to líbí ještě někomu jinému, má možnost mé rozšíření použít s vědomím, že chce-li psát přenositelné dokumenty, nebude rozšířené primitivy ani makra, která tyto primitivy volají, ve svém dokumentu používat.

Chce-li administrátor systému instalovat TEXovský formát včetně volby vhodného kódování, pak při inicializaci formátu může zavolat některou tabulku prvního typu a potom, těsně před povelem `\dump`, zakáže uživatelům sahat na kódovací vektory:

```
\let\xordcode=\undefined
\let\xchrcode=\undefined
\let\xprncode=\undefined
```

Tím má zaručeno, že uživatel nebude používat nestandardní rozšíření této implementace TEXu. Na veřejných sítích bych asi takové řešení doporučil. Tímto způsobem navíc vektory `xord` a `xchr` plní funkci, kvůli které je do TEXu už dávno zahrnul jeho autor: starají se o odstínění mezi kódovacími specifiky jednotlivých operačních systémů a pevně zvoleným vnitřním kódováním TEXu.

Původní záměr autora TEXu spočíval v tom, že nastavení kódovacích vektorů provede administrátor systému zásahem do změnového souboru `tex.ch` v době kompilace TEXu a tyto vektory budou pevně pro každou implementaci nastaveny. Bohužel, tato praxe se v případě implementací TEXu pro UNIX příliš nerozšířila, protože by to od administrátora vyžadovalo speciální aktivitu navíc, která

navíc pro nepoučeného nebyla zcela triviální. Zavedení primitivů `\xordcode` a `\xchrcode` a tabulek prvního typu posunuje úpravu kódovacích vektorů až na dobu inicializace formátu, ale také podstatně zjednodušuje jejich nastavení. To bych považoval za výhodu mého řešení. Já jsem vlastně neudělal nic jiného, než že jsem z pozice administrátora systému pozměnil změnový soubor `tex.ch` a změnu jsem udělal tak, aby výsledek mé práce byl pokud možno flexibilní.

Můžeme si položit otázku, proč profesor Knuth už dávno primitivy sahající k vektorům `xord` a `xchr` nezavedl. Zřejmě chtěl, aby se všechna TEXovská makra chovala zcela stejně na všech implementacích. Přímý přístup k vektorům `xord` a `xchr` byl pro něj v takovém případě nepřijatelný. V konstrukcích typu `\ifnum\xordcode‘@=‘@` se totiž může makro větvit podle toho, zda hostitelský operační systém používá stejné kódování jako ASCII nebo ne.

Poznamenejme, že se autorovi TEXu nepodařilo zcela zajistit stejné chování maker na všech implementacích. Když si nechám vypsat do souboru znak, který je v jednom operačním systému tisknutelný a v druhém ne, pak mám dva různé výsledky: buď přímo znak nebo `^^`*kód*. Když znovu v druhém běhu TEXu takový soubor přečtu, ale nejprve nastavím `\catcode‘^=12`, pak se může mé makro větvit podle toho, zda je testovaný znak v dané implementaci tisknutelný nebo nikoli.

Závěrečná poznámka

Každý si může pro své potřeby upravit TEX přímo ze zdrojového kódu. Viz též úvodní motto. Přitom je to úkol jednodušší, než by se na první pohled mohlo zdát. Mě osobně stačilo jeden večer listovat v [1] a vše si důkladně rozmyslet. Pak jsem druhý den dopoledne dostal myšlenku do počítače a vše vyzkoušel a odpoledne jsem napsal tento článeček. A věc je hotova. To vše díky velmi dobře dokumentovanému programu TEX.

Reference

[1] Donald E. Knuth. *TEX: The Program*, volume B of *Computers & Typesetting*. Addison-Wesley, Reading, MA, USA, 1986.

[2] Petr Olšák. *encTEX*, balík rozšiřující TEX volně k dostání na `ftp://math.feld.cvut.cz/pub/olsak/enctex`

[3] Libor Škarvada. Záplata rozšiřující TEX volně k dostání na `ftp://ftp.cstug.cz/pub/tex/local/cstug/skarvada`

České uvozovky

Tomáš Hála

Úvod aneb popis situace

Nedávno jsem byl nucen připravit do tisku jednu knihu, která se skládala převážně z vyprávění mnoha osob a z řady úryvků z denního tisku [8]. V přímé řeči i v úryvcích se opět vyskytovala přímá řeč či jiné úseky textu běžně zapisované v uvozovkách.

Tento problém není ve své podstatě složitý, neboť vnitřní uvozovky (uvozovky druhého řádu) se v češtině obvykle zapisují pomocí tzv. jednoduchých uvozovek (‚‘). Situaci však zkomplikovaly dvě skutečnosti:

1. Hrubý text nebyl připraven dostatečně přesně (chybějící uvozovky, neoznačené citáty apod.).
2. Do posledního okamžiku nebylo rozhodnuto, zda přímá řeč a citáty nejvyšší úrovně budou vloženy v uvozovkách, nebo budou označeny jiným způsobem (odsazení, kurzíva), což ovlivňuje zpravidla také tvar uvozovek nižší úrovně.

Ponechat tento problém bez řešení by znamenalo snížit efektivitu práce s TEXem na úroveň jistých WYSIWYG systémů, kde požadavek jakékoliv změny tohoto druhu má za následek „ruční“ opravu každého výskytu v celém textu. Pro ilustraci — v uvedené publikaci se uvozovky (první nebo druhé úrovně) vyskytují celkem 185×.

Literární přehled

Čeština

O uvozovkách se můžeme dočíst například v Pravidlech českého pravopisu [4]. Informace se však týkají pouze příkladu použití uvozovek v různých případech. Druhy uvozovek jsou popsány výčtem hned v první větě § 204: *Do uvozovek („“ ‚‘ »«) dáváme ty části projevu, ...*

Nová Pravidla českého pravopisu [5] uvádějí v § 137 totéž. Na konci tohoto paragrafu je však poznámka týkající se vnořenosti uvozovek: *Jestliže ve výraze v uvozovkách je třeba užít další uvozovky, lze využít jejich různé typy („“ ‚‘ »« aj.), ...*

V České mluvnici [2] (§ 36, odst. 4) jsou uvedeny pouze uvozovky první úrovně („“), v textu však jsou v první úrovni používány i uvozovky ‚‘ , zpravidla pro vysvětlení jiného významu slova (např. str. 75: *... naoko (‚zdánlivě‘) ...*).

Z těchto faktů i z příkladů uváděných v obou vydáních Pravidel lze vyvodit pro češtinu následující závěry:

1. Nejvyšší prioritu mají uvozovky typu „“.
2. Vnořené uvozovky se připouštějí, přičemž počet vnoření není omezen.
3. U dvojic ‚‘ a »« není stanoveno pořadí, použití dvojice ‚‘ v 2. úrovni je však běžné.
4. V případě potřeby lze použít i jiné dvojice znaků pro kteroukoliv úroveň.

Slovenština
O uvozovkách hovoří také ve slovenská Pravidla [6] (kapitola 2.7., str. 113): *Úvodzovky („“) sa píšu: ...* V poznámce v téže kapitole je potom připuštěna i varianta ‚‘ pro první úroveň, což však platí pouze pro tištěnou podobu. Stejná dvojice (‚‘) se ve slovenštině smí použít pro označení citátu uvnitř jiného citátu. Jiné druhy uvozovek nejsou uvedeny. Vícenásobné vnoření rovněž není vyloučeno.

Angličtina
Anglické uvozovky jsou tvořeny znaky “”. V tisku se připouští i dvojice ‘’ ([1], § 522).

Ruština
V rusky psané literatuře (např. [3], [7]) se používají znaky « » nebo originální ruské «».

Sada makropříkazů `uviq`[1]

Všechny výše uvedené informace a závěry jsem se pokusil využít při tvorbě sady makropříkazů pro použití uvozovek. Systém byl odladěn v prostředí LaTeXu verze 2.09 na počítači PC 486DX4/100 s operačním systémem MS-DOS verze 6.20.

Makropříkazy této verze jsou nadstavbou stylu `czech`, verze 1.141, a proto v současné době je snaha vložit tuto sadu makropříkazů přímo do stylu `czech`, případně `slovak`.

Makropříkaz `\uviq`
Základem je makropříkaz `\uviq` s jedním parametrem, který nahrazuje dobře známý makropříkaz `\uv` z českého či slovenského stylu, jeho použití je tudíž

[1] UVozovky s vyšším IQ.

stejné. `\uviq` pracuje nad čítačem `\@uvLevel`: při každém volání se zvyšuje hodnota čítače o jedničku, tudíž tělo makropříkazu `\uviq` rozpozná úroveň vnoření, a proto může nastavit vhodný znak pro otevření uvozovek. Na konci makropříkazu `\uviq` je volán makropříkaz `\closeuviq`, který je obdobou makropříkazu `\closeuv`. Tento uzavírací makropříkaz jednak vybere vhodný znak pro uzavření uvozovek na základě hodnoty v čítači `\@uvlevel`, a jednak hodnotu v čítači sníží o jedničku. Čítač tedy v průběhu práce s „inteligentními“ uvozovkami soustavně udržuje hodnotu úrovně vnoření uvozovek.

■ Příklad:

Diskutující řekl: „Ve Zpravodaji bylo napsáno: ‚Máme »inteligentní« uvozovky!‘ “

```
Diskutující řekl: \uviq{Ve
Zpravodaji bylo napsáno:
\uviq{Máme \uviq{inteligentní}
uvozovky!}}
```

□

Nastavení dvojic znaků

Existuje celkem 5 dvojic identifikátorů (pro 5 různých úrovní uvozovek), pod kterými jsou nebo mohou být definovány příslušné znaky. Tyto identifikátory jsou zvnějšku uživateli nepřístupné. Vlastní nastavení se proto musí provést makropříkazem `\UVIQ`, který má tři parametry: číslo úrovně (1–5), popis otevírací uvozovky, popis uzavírací uvozovky. Makropříkaz `\UVIQ` nelze použít v parametru makropříkazu `\uviq`. Nepovažuji za rozumné měnit nastavené znaky při „otevřených“ uvozovkách.

■ Příklad — implicitní nastavení pro češtinu:

```
\UVIQ{1}{\@clqq}{\@crqq}
\UVIQ{2}{\@clq}{\@crq}
\UVIQ{3}{\@clqqq}{\@crqqq}
```

□

Nastavení nejvyšší přípustné úrovně uvozovek

Z předchozí části vyplývá, že lze pracovat až s 5 různými úrovněmi uvozovek, avšak počet úrovní lze samozřejmě rozšířit podle potřeby. V praxi (a ve shodě s názory účastníků konference `csTeX`) se však nepoužívají více než tři úrovně. Proto byl zaveden čítač `@uvMaxLevel`, do něhož lze nastavit nejvyšší přípustnou hodnotu úrovně uvozovek. Při každém volání makropříkazu `\uviq` se porovnává hodnota tohoto čítače s aktuální hodnotou čítače `@uvLevel`. Byl-li makropříkaz `\uviq` volán vícekrát, než je přípustné, je při překladu ohlášena chyba.

Hodnotu čítače `@uvMaxLevel` lze nastavit pouze ve stylových souborech.

Implicitní nastavení a přepínání mezi různými jazyky
Po připojení této sady makropříkazů k dokumentu se jako aktivní nastaví hodnoty pro češtinu. Nastavení pro jiné jazyky získáme makropříkazem `\uviqInit`, kde jako parametr stojí dvoupísmenný kód země: pro slovenštinu `\uviqInit{sk}`, pro angličtinu `\uviqInit{uk}`, pro ruštinu `\uviqInit{ru}`. České nastavení lze obnovit makropříkazem `\uviqInit` s parametrem `cz`.

Nastavení pro ruštinu je připraveno pro znakovou sadu `wncyr`. Použití obdobných znaků ze sady `csr` lze zajistit inicializačním makropříkazem `\uviqInit{rucz}`.

Makropříkaz `\uviqInit` definuje pro každý jazyk příslušné dvojice znaků, jak bylo uvedeno na str. 119 a dál, včetně nejvyšší přípustné úrovně. Počet nastavených dvojic odpovídá nejvyšší přípustné úrovni, znaky v ostatních dvojicích jsou definovány jako prázdné makopříkazy.

Vztah mezi `\uviq` a `\uv`
Makropříkaz `\uviq` nebrání použití makropříkazu `\uv` ze stylů `czech` či `slovak`.

Závěr

Předložená sada makropříkazů řeší problém nastíněný v úvodu. Řešení je zobecněno, a proto umožňuje uživateli provádět vlastní úpravy sazby uvozovek. Jaké dvojice si uživatel nastaví a kolik úrovní použije, záleží pouze na jeho vkusu. V každém případě by však tyto změny neměly být provedeny na úkor přehlednosti zpracovávaného textu.

Vzhledem k omezeným časovým možnostem jsem nebyl schopen zpracovat více jazyků. Proto budu vděčen za další podněty i informace týkající se uvedeného problému.

Literatura

[1] Hais, K.: Anglická mluvnice. 363 s. SPN, Praha 1991.
[2] Havránek, S. — Jedlička, A.: Česká mluvnice. 592 s. SPN, Praha 1988.
[3] Kipling, R.: Poems. Short stories. 457 s. Raduga, Moskva 1983.
[4] Pravidla českého pravopisu. Školní vydání. 17. vydání. 425 s. SPN, Praha 1989.
[5] Pravidla českého pravopisu. 1. vydání. 391 s. Academia, Praha 1993.
[6] Pravidlá slovenského pravopisu. 1. vydanie. 533 s. Veda, Bratislava 1991.
[7] Rusko-slovenský a slovensko-ruský technický slovník. 1136 s. Alfa, Bratislava 1984.
[8] Stejskalová, S.: Kamelot: Návrat na místo činu. 178 s. Konvoj, Brno 1997.

LyX: téměř WYSIWYG prostředí pro LaTeX

Petr Mejzlík

LyX je volně dostupný textový procesor pro X-Windows, který vypadá a funguje obdobně jako jiné textové procesory (je nápadně podobný MS-Wordu), ovšem používá LaTeX pro sazbu svých dokumentů. Nejde o plně WYSIWYG program: v době editování není vidět například zalomení řádků a stránek nebo výsledná čísla u formulí nebo citací. Všechny tyto podrobnosti se dozvíme až po vysázení dokumentu do .dvi nebo .ps podoby (obojí provádí interně LyX tak, že nejprve převede dokument do LaTeXu a poté spustí `latex` a `xdvi`, případně také `dvips` a `ghostview`). To je ale nezbytné (alespoň v principu) až při finálním ladění grafické podoby dokumentu, protože informace, které nesouvisejí s rozměry papíru nebo s jinými kvantitativními parametry sazby, poskytuje samotný LyX.

Podle mého názoru v sobě LyX šťastně spojuje snadnost použití běžných textových procesorů (ve kterých nelze udělat *syntaktickou* chybu) s šířkou vyjadřovacích prostředků a stabilitou LaTeXu. Pomocí menu a klávesových maker jsou dostupné běžné typografické položky, jako je hierarchické členění textu, volba fontu nebo vkládání tabulek, obrázků a poznámek pod čarou. V LyXu lze podobně jako v LaTeXu používat symbolická návěští a citace zpracovávané BibTeXem. Silnou stránkou LyXu je editace matematických formulí, dovolující vytvářet formule jak pomocí myši, tak i použít standardní makra (například napíšeme-li „`\frac␣`“, zobrazí se zlomková čára). Pokud se ve své snaze o co nejlepší výsledek dostaneme za hranice možností LyXu, můžeme vždy vsunout úsek LaTeXovského kódu, který je pak beze změny vložen do produkovaného souboru.

1. Použitelnost LyXu

Je zřejmé, že LyX může být užitečný pro ty, kdo se nechtějí učit LaTeX, ale přitom by rádi využívali alespoň část jeho možností. Domnívám se ale, že LyX má smysl i pro uživatele, kteří LaTeX nebo Plain TeX již dávno zvládli, což, jak předpokládám, je případ většiny čtenářů Zpravodaje. Rád bych proto zde uvedl argumenty ve prospěch univerzálnosti LyXu, zároveň s námitkami (text v kurzívě), za které děkuji Petru Sojkovi.

- *V textovém editoru lze psát (resp. sázet) stejně rychle jako v LyXu.* To je myslím pravda tehdy, když velmi dobře znám LaTeX (tím si po sedmi letech praxe stále ještě nejsem jist), sázený text mám v jisté podobě připravený a zhruba vím, jak ho chci vysázet. Pokud ale píši text, který nemám předem

připravený a používám v něm komplikovanější typografické záležitosti, jako třeba matematiku, tabulky nebo obrázky, musím jejich převodu do LaTeXu věnovat tolik pozornosti, že nejsem schopen zároveň přemýšlet o obsahu. V LyXu mohu snadno psát i složitější matematické formule, aniž bych je měl před sebou na papíře.

- *Při psaní v LyXu se ztrácí jedna z hlavních výhod LaTeXu, kterou je logické značkování (tj. makra).* Je jistě lepší, když úseky textu, které chceme jistým speciálním způsobem zvýraznit, zapíšeme pomocí vhodně pojmenovaného makra a teprve nakonec se definitivně rozhodneme, jak má být toto makro definováno (například jestli má odpovídat kurzívě, tučnému písmu nebo kapitálkám). Nicméně u některých druhů dokumentů, například u článků určených pro odborné časopisy, můžeme taková rozhodnutí zpravidla udělat předem. Navíc se v LyXu dají libovolně používat LaTeXovská makra pomocí „TeX módu“, takže jejich (ne)používání je čistě věcí osobní disciplíny. Makra se dají definovat nebo načíst v hlavičce dokumentu, ke které se v LyXu dostaneme pomocí `Layout/LaTeX Preamble`.
- *Stylové soubory pro příspěvky na konference a do časopisů jsou určeny pro (La)TeX a ne pro LyX.* Pokud styl jen redefinuje standardní makra a nastavuje nejrůznější rozměry, stačí jej načíst v hlavičce (LaTeX Preamble). Jestliže ale styl definuje nová makra, která se v textu často vyskytují (například speciální matematická prostředí), je neustálé přepínání do TeXovského módu a zpět dost nepohodlné a navíc výsledek uvidíme stejně až v DVI souboru. V takovém případě je lepší napsat text v LyXu bez použití nových maker, exportovat ho do LaTeXu a dokončit v této podobě. Alternativním řešením je sestavení stylového souboru pro LyX (tyto soubory mají příponu `.layout`). Součástí distribuce LyXu jsou stylové soubory pro standardní třídy dokumentů LaTeXu (article, book, letter, report) a navíc styl amsart, který nastavuje charakteristiky sazby na hodnoty definované v $\mathcal{AMS}$-LaTeXu. LyX ale zatím neimplementuje rozšíření matematického módu $\mathcal{AMS}$-LaTeXu, s výjimkou `\cases`.
- *LyX zatím není moc rozšířen, a proto nevhodný pro výměnu dokumentů (dokument obsahuje zbytečné LyXové věci navíc).* Pokud chci svůj text psaný v LyXu předat někomu, kdo LyX nepoužívá, tak mi opravdu nezbývá než jej konvertovat do LaTeXu nebo do jiného formátu. Na druhé straně je výsledný LaTeXovský dokument přehledný a dá se bez problémů editovat. V posledních verzích do něj již LyX nevkládá definice žádných speciálních maker, s výjimkou svého loga.

Pro některé záměry, jako je například finální sazba profesionálních publikací, není LyX patrně přínosem. Podle mého názoru ale znatelně zvyšuje efektivitu samotného psaní, obzvlášť u textů s matematickými formulemi. Nevýhodou současné verze LyXu je absence prostředku pro převod z LaTeXu do LyXovského

formátu. Již napsané LaTeXovské dokumenty je proto lepší doplňovat nebo aktualizovat v původním tvaru. Konverze z LaTeXu nicméně má být v dohledné době implementována.

2. Instalace

Hardware a operační systém. LyX lze provozovat pod libovolnou variantou unixu s X-Windows nebo pod OS/2 s XFree (volně dostupná portabilní verze X-Windows). Pod Linuxem postačuje 486 s 8MB RAM, i když pro práci s rozsáhlejšími texty se vyplatí rozšířit paměť na 16MB.

Distribuce LyXu. Distribuční balíky LyXu jsou primárně dostupné pomocí anonymního ftp na adrese `ftp://ftp.via.ecp.fr/pub/lyx/`, stejné soubory lze najít i na `ftp://la1ad.uio.no/pub/lyx/` a na dalších serverech. Najdete zde velké množství verzí LyXu, z nichž většina je určena především pro samotné vývojáře (jsou to tedy alfa verze). Obecně platí, že liché verze LyXu (0.9, 0.11, 0.13) jsou alfa verze, zatímco sudé verze (0.8, 0.10, 0.12, 0.14) jsou beta verze, tj. jsou určeny pro veřejnost. Zatím poslední beta verze s číslem 0.10.7 má ještě problémy s podporou jazyků se znakovou sadou odlišnou od ISO Latin-1. V době, kdy tento článek vyjde, bude ale snad již hotová beta verze 0.12, která zmíněný nedostatek odstraňuje. Navíc nejnovější současné alfa verze (0.11.36 a vyšší) jsou již dostatečně stabilní a je v nich implementováno téměř vše, co bude ve verzi 0.12, včetně mezinárodní podpory. Pro výběr verze pro instalaci proto platí následující pravidla:

(i) Zvolte poslední beta verzi, pokud má číslo alespoň 0.12.0.

(ii) Pokud dosud není k dispozici verze 0.12.0, pak můžete zvolit poslední verzi ze série 0.11 (na `ftp.via.ecp.fr` ji najdete v adresáři `/pub/lyx/devel/0.11/`). Pokud si chcete být jisti, že daná verze je stabilní, pošlete mi mail.

Z webové stránky Davida Johnsona `http://www.lehigh.edu/~dlj0/LyriX.html` se dá stáhnout hotová binární verze LyXu 0.10.7 pro Linux a AIX. Pravděpodobně zde bude také umístěna binárka verze 0.12.0, jakmile bude k dispozici. Staticky sestavená binární verze není sice optimální pokud jde o využití operační paměti (nepoužívá totiž sdílené knihovny), je ale ideální pro první seznámení s LyXem, protože její instalace nevyžaduje prakticky žádný čas ani speciální znalosti.

Překladač a knihovny. Pokud nepoužijete binární distribuci LyXu, potřebujete kvalitní překladač C++, například gcc verze 2.7.2.1 a vyšší. Dále je třeba instalovat knihovnu Xforms verze 0.86 nebo 0.87 a přesvědčit se, že máte v systému knihovnu Xpm, nejlépe verze 4.7 nebo vyšší (to je totéž jako Xpm verze 3.4g,

protože tato knihovna má z historických důvodů dvojí číslování). U Xpm obvykle postačuje verze, která byla instalována spolu s operačním systémem. Aktuální verze knihovny Xforms se dá najít buď na distribučním uzlu LyXu (`ftp://ftp.via.ecp.fr`) v adresáři `/pub/xforms/`, nebo přímo u zdroje na `http://bragg.phys.uwm.edu/xforms/`. Nová verze Xpm se dá v případě potřeby získat na `ftp://ftp.x.org/contrib/libraries/`. Pokud si nejste jisti, jestli máte ve svém systému nainstalovány zmíněné knihovny, stačí si po rozbalení distribučního balíku přečíst výstup programu `configure` (program je součástí distribuce LyXu).

Překlad a konfigurace LyXu. Pokud překládáte LyX ze zdrojových textů, je vhodný následující postup (při instalaci z binární distribuce odpadají první tři kroky):

(i) Rozbalte distribuční sadu, přečtěte si soubory `README` a `INSTALL`, pusťte `./configure --help` v hlavním adresáři distribuce.

(ii) Spusťte `./configure` s parametry určujícími adresáře pro instalaci a další volby. Pokud neuvedete žádné parametry, bude program `lyx` nainstalován do adresáře `/usr/local/bin/` a soubory s ním související do adresáře `/usr/local/share/lyx/`. Pokud ale například máte knihovnu Xforms umístěnou na nestandardním místě, třeba v `$HOME/xforms/FORMS/`, spustíte `./configure` s parametry
`--with-extra-lib=$HOME/xforms/FORMS/`
`--with-extra-inc=$HOME/xforms/FORMS/`.

(iii) Pokud konfigurace proběhla bez chybových hlášení, spusťte `make all` a `make install`.

(iv) Spusťte LyX: tím se ve Vašem home adresáři vytvoří podadresář `.lyx`, který se dá používat pro osobní konfiguraci LyXu, odlišnou od systémových konfiguračních souborů uložených standardně v adresáři `/usr/local/share/lyx/`.

(v) Upravte konfigurační soubor (buď přímo systémový `lyxrc` nebo `~/.lyx/lyxrc`) tak, aby vyhovoval Vašim potřebám.

3. Čeština

LyX umožňuje psát česky bez jakýchkoliv úprav. Stačí, když v `Options/Keyboard` zvolíme českou klávesnici: česká písmena, která nejsou v Latin-1 fontech, jsou pak graficky aproximována. Problémy ale nastanou, pokud chceme editovat dokument exportovaný do LaTeXu: písmena s akcenty jsou v něm zapsána pomocí TeXovských maker (například `\v{e}` namísto ě). Nepodaří se nám také korektně načíst textové soubory psané v ISO Latin-2 kódování. Pokud tedy chceme LyX běžně používat pro psaní českých textů, je potřebné tyto

nedostatky odstranit. Řešení spočívá především v instalaci ISO Latin-2 fontů pro X-Windows.

Fonty. V současné době je již k dispozici celá řada X-Windows fontů pro znakovou sadu Latin-2. Nejlepším místem, kde se dají sehnat, je webová stránka Primoze Peterlina `http://sizif.mf.uni-lj.si/linux/cee/iso8859-2.html`. Pro LyX ale potřebujeme obrazovkové fonty v mnoha různých velikostech, čímž se výběr prakticky zužuje na postscriptové fonty. V současné době patrně jediné veřejně dostupné fonty tohoto typu jsou dílem Pétera Soóse z Řádu sv. Benedikta v maďarské Pannonhalmě. Jsou dostupné pomocí anonymního ftp v souboru `ftp://ftp.osb.hu/pub/misc/fonts/local/latin-2/l2pfb004.zip`. Stačí rozbalit tento `.zip` soubor do některého adresáře (například do `/usr/X11R6/lib/X11/fonts/osbfonts/`) a adresář přidat do cesty, ve které se hledají fonty tím, že napíšeme „`xset +fp adresář`“ do souboru `$HOME/.xinitrc` a restartujeme X-Windows. Po spuštění X-Windows se můžeme příkazem `xset q` přesvědčit, jestli je cesta správně nastavena. Adresář můžeme přidat i přímo do systémové konfigurace X-Windows: v Linuxu stačí napsat „`FontPath adresář`“ před první `FontPath` řádek v souboru `/usr/lib/X11/XF86Config`.

Volby týkající se fontů pak budou v `lyxrc` vypadat následovně:

```
\roman_font "-*-times new roman"
\sans_font -*-arial
\typewriter_font "-*-courier new"
\font_norm iso8859-2
```

Klávesnice. Česká klávesnice s americkou alternativou se v `lyxrc` nastaví pomocí

```
\kbmap true
\kbmap_primary czech
\kbmap_secondary american
```

Pokud ovšem máte českou klávesnici nastavenou prostředky X-Windows (to je možné v X11R6), nemusíte `\kbmap*` používat.

Osvědčilo se mi také nastavení mazání směrem dozadu a použití `Pause` pro přepínání klávesnice: přepíná postupně mezi primární klávesnicí (v našem případě česká), sekundární klávesnicí a stavem, kdy LyX do mapování klávesnice nijak nezasahuje. Ve standardním `lyxrc` stačí odkomentovat následující řádky:

```
\bind "Delete" "delete-backward"
\bind "Pause" "keymap-toggle"
```

LaTeX. Pro korektní sazbu českých textů je vhodné v `lyxrc` nastavit následující parametry:

```
\fontencoding T1
```

`\inputencoding latin2`
Použití Babelu pro zpracování češtiny se v `lyxrc` zapíná příkazem
`\language czech`
Babel definuje také „německé“ uvozovky, které jsou nerozlišitelné od českých. V `lyxrc` tomu odpovídají příkazy
`\quotes_language german`
`\quotes_times 2`
Pokud pro češtinu používáte namísto Babelu speciální format file, například `cslatex.fmt`, je třeba namísto těchto voleb nastavit
`\latex_command cslatex`

4. Další informace

Přehledné údaje o LyXu, stejně jako plány na jeho další vývoj lze najít na následujících webových adresách:
`http://la1ad.uio.no/~larsbj/lyx.shtml` (Lars Bjønnes),
`http://www.lehigh.edu/~dlj0/LyriX.html` (David Johnson),
`http://www-pu.informatik.uni-tuebingen.de/users/ettrich` (Matthias Ettrich, zakladatel projektu).

Mailing list pro uživatele: přihlásíte se tím, že pošlete mail na adresu `lyx-users-request@fiwi02.wiwi.uni-tuebingen.de` a do subjectu napíšete `subscribe`

Veškeré zprávy této diskusní skupiny jsou archivovány na `http://bioclox.bot.biologie.uni-tuebingen.de/mailing-archive/lyxlist/maillist.html`

Mailing list pro vývojáře LyXu má odlišnou proceduru pro přihlašování: pošlete zprávu na `majordomo@via.ecp.fr` a do jejího *těla* napíšete
`subscribe lyx`
`end`
Archiv je dostupný na `http://www.via.ecp.fr/~andre/lyx/archive`.

Petr Mejzlík
`mejzlik@fi.muni.cz`

Putování písmene ř z klávesy na papír

Petr Olšák

V tomto článku podrobně rozeberu, co se odehraje v počítači mezi zmáčknutím písmene ř na klávesnici a vytištěním písmene ř na papíře. Ono písmeno je v jednotlivých fázích zpracování interpretováno jako číselný kód. Ukážeme, že se zdaleka nemusí jednat o kód jediný. V průběhu zpracování může písmeno ř (a ostatní písmena a znaky, které používáme) absolvovat až sedm různých kódování, přičemž v každém z nich může (ale nemusí) být naše písmeno reprezentováno jiným kódem.

Článek by měl dokumentovat velkou flexibilitu přístupu k různým kódováním při zpracování dokumentu TEXem. Uvidíme, že možností je mnoho a těžko bychom hledali u jiného softwaru obdobu.

Putování písmene ř rozdělíme do několika fází. V každé fázi může být tento znak reprezentován jiným kódem. Jednotlivé fáze můžeme zhruba načrtnout do následujícího schématu:

klávesa ř	
K-kódování	… hardwarové kódování klávesnice
↓	… ovladač klávesnice
A-kódování	… kódování aplikace textového editoru
↓	… uložení dokumentu na disk
D-kódování	… kódování dokumentu na disku
↓	… input preprocesor TEXu
I-kódování	… kódování dokumentu na vstupní straně maker
↓	… expanze maker
T-kódování	… kódování na úrovni hlavního procesoru a v dvi
↓	… expanze dvi podle virtuálních fontů
F-kódování	… kódování fontu, nebo jeho Encoding vektoru
↓	… Encoding vektor
P-kódování	… vnitřní kódování PostScriptového fontu
tištěné ř	

Uvedeme seznam nejběžnějších kódování, která se na úrovni jednotlivých fází zpracování obvykle v českém a slovenském jazyce používají:

K-kódování … scan kódy klávesnic
A,D,I-kódování … ISO-8859-2, Kamenických, PC-Latin2=CP852, CP1250, …
I,T,F-kódování … kódování $\mathcal{CS}$-fontů, T1=Cork, …
F,P-kódování … Adobe StandardEncoding, …

V dalším textu se jednotlivými fázemi zpracování dokumentu budeme věnovat podrobněji.

K-kódování ⟶ A-kódování

Stisknutí klávesy se reprezentuje číselným kódem, který je odeslán do počítače. Například na klávesnicích PC se jedná o tzv. „scan kódy“, kde klávesy mají vesměs čísla podle svého rozmístění. Klávesa Esc má číslo jedna, klávesa A číslo 30, S 31, D 32 atd. V počítači pracuje ovladač klávesnice, který jednotlivou posloupnost stisků kláves interpretuje a běžící aplikaci již vrací aplikační kód. Je-li onou aplikací textový editor (což je při přípravě dokumentu pro TEX obvyklé), pak aplikace zobrazí znak na obrazovce a uloží si jej do paměti. K tomu zobrazení na monitoru potřebuje použít font, který musí mít rovněž A-kódování, jinak bychom byli zmateni.

Přehled o tom, co kdy bylo zmáčknuto a uvolněno a co to v dané situaci znamená, musí udržovat ovladač klávesnice. Například stisk levé klávesy Shift vyvolá scan kód 42. Pokud uživatel klávesu drží, pak ovladač musí následující klávesy interpretovat odlišným způsobem. O puštění klávesy Shift se ovladač dozví podle scan kódu 42+128. Ovladač klávesnice obvykle pracuje v několika režimech (český/standardní). Naše klávesa ř samostatně neexistuje. Písmeno ř může být v českém režimu vyvoláno buď použitím smluvené klávesy (obvykle klávesa 5/%) nebo stiskem a uvolněním smluvené „mrtvé klávesy“ (obvykle klávesa +/=) a následným stiskem klávesy r. Vše může vypadat i jinak. Záleží na tom, jak je ovladač klávesnice naprogramován.

V UNIXu je scan kód z klávesnice zpracován rovněž ovladačem klávesnice (v jádru operačního systému) na tzv. key kód. Pokud běží X Window System, pak je key kód obvykle zpracován X aplikací podle globální mapovací tabulky, společné všem aplikacím. Tuto tabulku je možné měnit programem `xmodmap`. Aplikace může také převzít inteligenci ovladače klávesnice do svých rukou a starat se o přepínání režimů klávesnice a transformaci znaků. Mám na mysli třeba Emacs při použití maker z balíku `emacs-czech` [EC]. Kód znaku se tedy může měnit na třech místech: v jádru systému, na úrovni globální mapovací tabulky a uvnitř konkrétní aplikace. Ve všech případech existují konfigurační možnosti, jak do konverzního algoritmu zasáhnout.

Font, který používá aplikace k zobrazení právě stisknutého znaku, je obvykle fontem, který je nějakým způsobem instalován v operačním systému. A-kódování

tedy volíme shodně s kódováním fontu v operačním systému. U jiného druhu textového zpracování tím veškerá starost o kódování končí (např. MS Word a jiné), protože pro tisk dokumentu používá aplikace rovněž fonty, které nabízí operační systém.

My uživatelé TEXu víme, že mezi fontem použitým v operačním systému a fontem určeným pro kvalitní typografické zpracování sazby může být propastný rozdíl. Font operačního systému volíme takový, aby hezky vypadal a byl dobře čitelný na obrazovce, zatímco pro tisk volíme font, který nejlépe odpovídá typografickému záměru, kvalitě papíru, tiskové technologii apod. Proto se pusťme do dalších fází zpracování dokumentu.

A-kódování ⟶ D-kódování

Jakmile požádáme aplikaci (textový editor), aby uložila dokument na disk, může aplikace při této činnosti spustit nějaký filtr, který může změnit kódování dokumentu. Například pro MS Windows existují (prý) editory, které vytvářejí a zobrazují dokument v CP1250, ale po uložení jej máme třeba v PC-Latin2. Vstupní bránu TEXu pak musíme připravit na to, že bude číst kódování v PC-Latin2, ačkoliv pracujeme s operačním systémem, který používá CP1250.

Obvykle při ukládání dokumentu kódování neměníme. V této fázi zpracování budeme měnit kódování asi jen ve zcela ojedinělých a opodstatněných případech.

Na druhé straně je běžnou praxí převádět dokumenty před zpracováním TEXem z jednoho kódování do druhého. Označme to jako D_1-kódování ⟶ D_2--kódování. Ve většině TEXovských instalací jsou konverzní utilitky k dispozici. Konverzi provádíme například tehdy, když obdržíme dokument v jiném kódování, než používáme v našem systému.

D-kódování ⟶ I-kódování

Tato konverze již probíhá uvnitř TEXu na úrovni jeho input procesoru. Ve zdrojovém kódu TEXu jsou implementovány vektory `xord` a `xchr`, které se starají o odstínění kódování použitého v hostitelském operačním systému od vnitřního kódování TEXu. Toto vnitřní kódování se předpokládá jednotné a nezávislé na kódování, které je podporováno hostitelským operačním systémem. To byl původní záměr autora TEXu. Knuth v sedmibitových dobách uvažoval existenci systémů s kódováním EBDIC, zatímco on volí v TEXu kódování ASCII. Na systémy s ASCII jsou tedy vektory `xord` a `xchr` nastaveny tak, že se chovají jako identické zobrazení, zatímco na těch druhých systémech dnes už vymřelých dinosaurů byly uvedené vektory příslušným způsobem pozměněny.

Je třeba poznamenat, že vektory `xord` a `xchr` se starají o oboustranné konverze, tj. nejen „dovnitř“ TEXu, ale též při výstupu na terminál a do logu (nikoli

do dvi). Proto TEX nabízí všechny své textové výstupy v kódování podle hostitelského operačního systému, ačkoli si sám interně může pracovat v jiném kódování.

Bohužel, nastavení překódovacích vektorů `xord` a `xchr` je možné jen před kompilací TEXu ze zdrojového souboru `web`. Za provozu TEXu lze měnit tyto vektory jen na některých implementacích. Tam, kde změna kódování za provozu není možná, je obvykle pevně nastaveno identické zobrazení.

Implementace TEXu zvaná emTEX (DOS, OS/2) je vybavena možností uvedené vektory měnit velmi volně prostřednictvím tzv. `tcp` tabulek. Tyto tabulky lze nejprve editovat, pak převést do binární podoby a předložit je TEXu v době inicializace formátu. Většinou se používají tabulky, které mají na vstupní straně kódování podle Kamenických, PC-Latin2 nebo CP1250 a na výstupní straně kódování $\mathcal{CS}$-fontů nebo T1 nebo expanze na TEXovské sekvence. Poslední možnost způsobí, že naše `ř` vstupuje do TEXu nakonec jako `\v r`. Tuto třetí možnost nedoporučuji, protože znemožňuje výskyt písmene ř ve verbatim prostředí.

UNIXové instalace TEXu, které vycházejí z implementace `web2c` a jsou ozáplatovány kódem podle `ftp://ftp.cstug.cz/pub/tex/local/cstug/skarvada` mají rovněž možnost měnit uvedené vektory `xord` a `xchr` za provozu TEXu, ovšem kódovací tabulku nemůžeme editovat, pouze si můžeme vybrat jednu z několika málo možností. Mezi zajímavou možnost patří zřejmě konverze ISO-8859-2 $\longrightarrow$ T1.

Existuje i jiná úprava TEXu, kterou jsem si udělal sám, a která umožňuje přímý přístup k vektorům `xord` a `xchr` za běhu TEXu [ENCTEX].

I-kódování $\longrightarrow$ T-kódování

Než se naše písmeno `ř` propracuje z input procesoru do hlavního procesoru TEXu, může s ním expand procesor pořádně zatočit. Této problematice se budeme nyní věnovat podrobněji.

V nejběžnějším případě, tj. když není vstupní písmeno `ř` aktivní, se nestane nic. Takže I-kódování je rovno T-kódování. Tak tomu je například v csplainu.

V LATEXu též implicitně nedochází ke konverzi těchto dvou kódování. Pokud se ale použije balík `inputenc.sty`, například:

```
\usepackage[cp852]{inputenc}
```

pak se všechny znaky s kódy nad 128 stávají aktivními a jejich význam je definován v souboru `*.def` (v našem příkladě `cp852.def`) prostřednictvím sekvence `\DeclareInputText`. Například pro písmeno `ř`, které je v CP852 kódováno jako `"FD` (hexadecimálně), najdeme v definičním souboru:

```
\DeclareInputText{"0FD}{\v r}
```

Znamená to, že náš znak `ř` se bude expandovat na `\v r`. Co se dále stane s touto sekvencí, záleží na volbě T-kódování a tím se budu zabývat za chvíli.

Ve stylu `inputenc.sty` je zajímavým způsobem řešeno definování znaku, který je aktivní, a jehož kód makro `\DeclareInputText` přečte ze svého parametru. Je k tomu použit primitiv `\uppercase` zhruba takto:

```
\def\DeclareInputText #1{\bgroup \uccode`~=#1
   \uppercase{\egroup \def~}}
```

Zápis

```
\DeclareInputText{"0FD}{\v r}
```

tedy expanduje na

```
\def ř{\v r}
```

přičemž ono `ř` je aktivní token kategorie 13 s kódem `"0FD`. Jak je to uděláno? Znaku `~` je přechodně (mezi `\bgroup` a `\egroup` přiděleno `\uccode` odpovídající načtenému parametru, takže primitiv `\uppercase` změní v prvním svém průchodu znak `~` na aktivní `ř`. V druhém průchodu se uzavírá skupina (`\egroup`) a otevírá definice `\def ř`.

V případě LaTeXu existuje jeden drobný zádrhel se zápisem `\'a` například pro písmeno `á`. Makro `\'` má totiž v LaTeXovém prostředí tabbing pozměněný význam. Kdyby uživatel v tomto prostředí použil písmeno `á` a ono by se expandovalo na `\'a`, pak by se uživatel asi nestačil divit. Balík `inputenc.sty` proto používá pro `á` expanzi na `\@tabacckludge'a` a LaTeXové jádro expanduje `\@tabacckludge'` na `\'` jen v případě, že zrovna nejsme v prostředí tabbing. Pokud v něm jsme, řeší LaTeX problém dlouhého á jinak.

V LaTeXu a jmenovitě v NFSS je velmi důkladně ošetřeno další chování sekvencí typu `\v r`, resp. `\'a`. To záleží na volbě T-kódování, kterému LaTeX říká `\fontencoding`. Je-li zvoleno kódování CM fontů, které se v LaTeXu jmenuje OT1, pak zmíněné sekvence expandují na `\accent20r`, resp. `\accent19a`. To je řečeno v souboru `ot1enc.def` na následujících řádcích:

```
\DeclareTextAccent{\'}{OT1}{19}
\DeclareTextAccent{\v}{OT1}{20}
```

T-kódování je vnitřní TeXovské kódování, ve kterém jsou načteny vzory dělení slov a ve kterém TeX vystupuje do `dvi`. Aby bylo možno použít české dělení slov, musí být T-kódování bezpodmínečně osmibitové. Předchozí příklad s kódováním OT1 je tedy pro češtinu nepoužitelný. Uvažujme proto například kódování $\mathcal{CS}$-fontů, kterému LaTeX říká IL2. V souboru `il2enc.def` je řečeno k našemu písmenu `ř` toto:

```
\DeclareTextComposite{\v}{IL2}{r}{248}
```

Pokud se použije font s kódováním IL2, pak bude sekvence `\v r` expandovat na znak s kódem 248. Podobných řádků, jako je tento, najdeme v souboru

`il2enc.def` více. LaTeX se do tohoto souboru podívá, pokud je nastaveno `\fontencoding` na IL2. Uživatel může toto kódování nastavit takto:

```
\usepackage[IL2]{fontenc}
```

Jak je makro `\DeclareTextComposite` definováno, nebudu pro nedostatek místa rozvádět. Zkuste si nastavit `\tracingmacros=2` a nechte zpracovat LaTeXem sekvenci `\v r` jednou v režimu OT1 a jednou třeba při IL2. V obou případech se po pohledu do `log` souboru zhrozíte, jak intenzivně a komplikovaně je tímto problémem zatížen expand procesor. Nicméně víme, že expanze nakonec při OT1 vede na `\accent20r` zatímco při IL2 se věc expanduje na samotný znak s kódem 248.

Aktivovat akcentované znaky použitím balíku `inputenc.sty` může mít smysl třeba tehdy, pokud chceme v průběhu jednoho dokumentu měnit I-kódování nebo T-kódování. Na oba případy jsou uvedená makra připravena. Při změně I-kódování (pro přepínání píšeme `\inputenc{KŌD}`) makro jednoduše předefinuje všechny aktivní znaky podle nového kódu. Při změně T-kódování (pro přepínání píšeme `\fontencoding{jinykod}`) se zase změní způsob expanze sekvencí `\v`, `\'` a podobných.

Chceme-li změnit T-kódování v rámci zpracování jediného dokumentu, nesmíme zapomenout současně přehodit vzory dělení slov, které jsou závislé na použitém T-kódování. Je tedy potřeba načíst před zpracováním dokumentu tolik variant vzorů dělení, kolik budeme používat různých T-kódování. Změna tohoto kódování uvnitř odstavce je ovšem velice problematická. Je sice možno uvnitř odstavce změnit vzory dělení, ale už není možno změnit tabulku `\lccode` jednotlivých znaků, která je interpretována globálně pro celý odstavec. Bohužel tato tabulka také ovlivňuje dělení slov. Je proto lepší T-kódování v průběhu sazby nestřídat a raději na různě kódované fonty navázat prostřednictvím virtuálních skriptů.

Je-li I-kódování $\neq$ T-kódování, pak je potřeba upravit konverzi znaků z malých na velké a naopak, na kterou dříve stačily primitivy `\uppercase` a `\lowercase`. Tyto primitivy totiž pracují podle `\lccode` a `\uccode` znaků *před* případnou expanzí posloupnosti tokenů. Přitom `\lccode` musí být nastaveno podle T-kódování, protože tyto kódy mají přímou souvislost s dělením slov. Hodnoty `\uccode` se z důvodu symetrie také nastavují podle T-kódování. Máme-li tedy I-kódování rozdílné, musíme nejprve provést expanzi a pak teprve použít primitiv `\uppercase` nebo `\lowercase`. Problém může řešit makro `\Uppercase`, které definujeme třeba takto:

```
\def\Uppercase #1{\edef\act{\uppercase{#1}}\act}
```

Uděláme si nyní malé shrnutí konverze I-kódování $\longrightarrow$ T-kódování v LaTeXu.

1a. Znak ř není aktivní: I-kódování = T-kódování.
1b. Znak ř je už v I-kódování rozložen na `\v r`: jdi na 2.

1c. Znak ř je aktivní (použitím `inputenc.sty`): jdi na 3.
2a. Jsme ve verbatim prostředí: vytiskne se `\v r`.
2b. Nejsme ve verbatim prostředí: `\v r` ⟶ T-kódování.
3. I-kódování ⟶ `\v r` ⟶ T-kódování (bez závislosti na verbatim prostředí).

Cílem tvůrců maker na konverzi I-kódování ⟶ T-kódování v LaTeXu je snaha po uživatelsky příjemnější přenositelnosti dokumentů na úrovni `*.tex` a oddělení problematiky vstupního kódování dokumentu od kódování fontů. Pokud každý „standardní“ LaTeXový dokument bude obsahovat volání stylu `inputenc.sty`, pak je vlastně v hlavičce řečeno, jaké vstupní kódování použil autor dokumentu. Dokument pak není závislý na kódování, které používá příjemce dokumentu. Příjemce se totiž nemusí starat o konverzi dokumentu do svého kódování, ale rovnou dokument LaTeXuje.

Tento záměr má ale k ideálu hodně daleko. Příjemce se totiž musí starat, jak jsou v dokumentu vyznačeny konce řádků. Pokud to nevyhovuje použitému operačnímu systému, musí provést před zpracováním konverzi dokumentu (o problémech s konci řádků viz [TBN] stranu 15). Může se také stát, že příjemce dostane dokument z elektronické pošty ve „svém“ kódování, zatímco odesílatel jej poslal v jiném „svém“ kódování — o konverzi se automaticky postaraly systémy pro přenos elektronické pošty. Pak zápis v hlavičce dokumentu o kódování odesílatele dokonce lže a může působit potíže. Snaha o co největší pohodlí uživatele je oprávněná právě u nástrojů, které umožňují přenos dokumentů z různých operačních systémů. V tomto případě jde o programy na práci s elektronickou poštou. Snažit se o totéž v samotném LaTeXu mi připadá nevhodné s tím, že to přináší více komplikací než užitku. Laického uživatele většinou vůbec nezajímá, jaké používá kódování. Takže se mu může jevit požadavek na uvedení tohoto údaje do hlavičky v `\usepackage[...]{inputenc}` jako zbytečnost, o kterou by se měl správně nainstalovaný systém postarat sám bez jeho pomoci. Používání tohoto balíčku také předpokládá, že A-kódování = D-kódování = I-kódování, což neguje původní Knuthův záměr, aby se rozdílnosti použitých operačních systémů řešily na úrovni D-kódování ⟶ I-kódování.

Pokud jsme po zpracování TeXem ztratili přehled o tom, jak dopadlo naše písmeno ř (to se vzhledem ke složitosti cesty od D-kódování po T-kódování může stát), pak je vhodné se podívat přímo do dvi souboru. Uděláme jednoduchý TeXový soubor, který tiskne do dvi pouze jediné písmeno `ř`. Pak použijeme program `dvitype`, který vypíše obsah dvi souboru v textové podobě. Najdeme tam například povel pro sazbu znaku písmene `ř` jako `set_1 248`.

I-kódování ⟶ T-kódování — další možnosti

Při popisu transformace mezi I-kódováním a T-kódováním nesmíme zapomenout na běžné situace, které vlastně dělají TeX TeXem. Jedná se o tyto možnosti:

(1) transformace mezi zápisem názvu makra a výsledným kódem v `dvi`, (2) transformace v matematickém módu podle `\mathcode` a (3) proměna skupin znaků na ligatury.

Transformace kontrolních sekvencí na výstup do `dvi` je řízena makrojazykem TEXu, který má v této oblasti velmi široké možnosti. Z důvodu stručnosti uvedeme jen jeden z mnoha příkladů. Sekvence `\alpha` na vstupu se při použití rodiny fontů Computer Modern převede na kód `"0B` (hexadecimálně) ve fontu `cmmi*`. Vysvětlit přesně, jak je to zařízeno, by bylo na delší vypravování, takže bez komentáře pouze zmíním, že to souvisí s touto deklarací:

```
\mathchardef\alpha="010B
```

V matematickém módu podléhají transformaci nejen kontrolní sekvence, ale i všechny běžné znaky. Napíšeme-li například `$-$`, pak toto „mínus“ se při použití rodiny Computer Modern převede na znak s kódem 0 ve fontu `cmsy*`, protože bylo v makrojazyku řečeno:

```
\mathcode`\-="2200
```

Nakonec zmíním přeměnu skupin znaků na ligaturu. Tato vlastnost je definována v metrice fontu `tfm`. Proměna v ligaturu se odehrává na úrovni T-kódování. Máme-li například na vstupu `---` a je zrovna použit font `cmr10`, pak se tyto tři znaky promění ve znak jediný s kódem `"7C`, tj. znak „—“.

T-kódování ⟶ F-kódování

Pod F-kódováním rozumíme kódování použitých fontů ve formátu PK, METAFONTu nebo Encoding vektoru PostScriptového fontu.

Je-li v `dvi` odkazováno na font, který je instalován jako `*.mf`, `*.pk` nebo PostScript a není přítomen skript virtuálního fontu `*.vf`, je T-kódování = F--kódování. To je dosti častý případ. Zde ale rozebereme případ, kdy mezi kódováním `dvi` souboru (T-kódováním) a F-kódováním stojí skript virtuálního fontu. Ten může obsahovat požadavek na změnu pozice znaku. Uvedeme abstraktní příklad. Nechť naše písmeno ř je v `dvi` na pozici 176 a má být vysázeno fontem `vcsr10`. Tento font je virtuální a odkazuje na font `csr10`, přitom pozici 176 mapuje na pozici 248. Pak bude nakonec naše písmeno ř sázeno fontem `csr10` z pozice 248. Uvedeme nyní, jak takový virtuální font `vcsr10` zhruba vypadá.

Především někde v instalaci musí být přítomen soubor `vcsr10.vf`. Je tam, kde ho najdou použité `dvi` ovladače. Chceme-li se mu podívat na zoubek, převedeme jej do čitelné podoby pomocí programu `vftovp`. Po konverzi najdeme v souboru `vcsr10.vpl` mimo jiné tyto informace:

```
(VTITLE Pouze priklad)
...
(MAPFONT D 0
  (FONTNAME csr10)
  )
...
(CHARACTER O 260
   (CHARWD R 0.391668)
   (CHARHT R 0.694445)
   (MAP
      (SETCHAR O 370)
      )
   )
```

Vidíme tedy, že znaky jsou implicitně čerpány z fontu `csr10` (viz `MAPFONT D 0`) a že pozice 260 (oktalově) je vysázena jako znak z implicitního fontu z pozice 370 (rovněž oktalově). Takto je možné překódovat i ostatní pozice. Podrobněji o virtuálních fontech — viz například [TST] stranu 75.

Na úrovni T-kódování ⟶ F-kódování může též dojít k „rozkladu“ našeho písmene ř na samostatné písmeno r a samotný háček. Prostřednictvím virtuálních skriptů lze `dvi` ovladači říci, ve kterém fontu hledat toto písmeno a akcent, a popsat přesný způsob, jak tyto segmenty sesadit k sobě. V naší předchozí ukázce virtuálního fontu by mohlo být napsáno:

```
(CHARACTER O 260
   (CHARWD R 0.391668)
   (CHARHT R 0.694445)
   (MAP
      (PUSH)
      (PUSH)
      (MOVERIGHT R -0.054167)
      (SETCHAR O 24)
      (POP)
      (SETCHAR C r)
      (POP)
      )
   )
```

Poznamenejme, že tato praxe „rozkladu“ znaku právě popsaným způsobem je hojně používána při použití standardních PostScriptových fontů v $\mathcal{C}\mathcal{S}$TEXu. U těchto fontů se totiž nepředpokládá, že v nich bude kresba písmene ř samostatně obsažena, protože třeba většina fontů z Adobe Type Library obsahuje pouze sadu znaků podle tzv. Adobe StandardEncoding. Tato sada má

písmena anglické abecedy a samostatné akcenty všeho druhu, takže všechna akcentovaná písmena naší abecedy lze prostě poskládat ze segmentů.

Pokud ale PostScriptový font obsahuje kompletní znak ř, pak není třeba provádět zmíněný rozklad na segmenty a nastupuje poslední fáze našeho překódovacího martyria:

F-kódování ⟶ P-kódování

Pod pojmem P-kódování rozumíme pořadí PostScriptových procedur ve fontu na vykreslení jednotlivých znaků. Toto pořadí je z hlediska sázecího systému naprosto nezajímavé, protože font je v PostScriptovém kódu použit vesměs prostřednictvím operátoru `show` (a alternativ). Tento operátor konvertuje pozici sázeného znaku na název procedury pro vykreslení požadovaného znaku. Konverze je definována prostřednictvím tzv. Encoding vektoru, který je součástí každého PostScriptového fontu. Obsahuje obvykle 256 jmen PostScriptových procedur. Vysvětlíme si to na příkladě.

Nechť PostScriptový font obsahuje proceduru na vykreslení kompletního znaku ř. Tato procedura má obvykle název `/rcaron`. Předpokládejme, že naše písmeno ř má v F-kódování pozici 248. Operátor `show` se při požadavku na vykreslení tohoto znaku podívá do Encoding vektoru na pozici 248 a měl by tam najít název `/rcaron`. To je název procedury, která je použita pro vykreslení požadovaného znaku. RIP nyní interpretuje proceduru `/rcaron` a znak vykreslí. Sláva, konečně vidíme znak ř po mnoha zákrutách na papíře!

Může se stát, že máme nějaký PostScriptový font s úplnou sadou české abecedy a že nám nevyhovuje jeho kódování. Jak vyplynulo z předchozího, není nic snadnějšího, než pozměnit tomuto fontu Encoding vektor. Ten najdeme v čitelné podobě například v souborech s příponou `pfb` nebo `pfa`. Před editováním je vhodné převést font z formátu `pfb` do `pfa` (například pomocí balíku `t1utils`, programy `t1binary` a `t1ascii`) a v tomto formátu provést požadované změny Encoding vektoru. Vlastní algoritmy procedur bývají většinou zašifrované, ovšem to nám nevadí. Nakonec převedeme font zpět do formátu `pfb` a máme jej připraven v požadovaném kódování.

Pokud je font přímo implementován v PostScriptovém RIPu a jeho sada znaků obsahuje úplnou českou abecedu, pak překódování takového fontu můžeme provést níže uvedenými povely, které zařadíme na začátek PostScriptového kódu. V tomto případě nelze přímo přepsat Encoding vektor, protože font je speciální případ tzv. „slovníku“, který je nastaven pouze ke čtení. Je proto potřeba tento slovník zkopírovat, v kopii pozměnit Encoding vektor a označit jej jako nový font pod jiným názvem. Například v RIPu máme font `Helvetica-E` s Encoding vektorem podle CP1250. My bychom ale chtěli používat tento font v kódování ISO-8859-2. Připravíme si proto Encoding vektor v našem kódování (kráceno):

```
/ISOLatin2Encoding [/.notdef /./notdef ...
  /space /exclam /quotedblright /numbersign /dollar /percent ...
  /at /A /B /C /D ...
  /rcaron /uring /uacute /uhungarumlaut /udieresis /yacute ...]
 def
```

a definujeme nový font s názvem `Helvetica-ISOLatin2`:

```
/Helvetica-E findfont
dup length dict begin
  {1 index /FID ne {def} {pop pop} ifelse} forall
  /Encoding ISOLatin2Encoding def
  currentdict
end
/Helvetica-ISOLatin2 exch definefont pop
```

Nakonec místo fontu `Helvetica-E` použijeme font `Helvetica-ISOLatin2`. Podrobněji o této technologii viz [RB].

Zásahy do fontu na úrovni F-kódování ⟶ P-kódování jsou nezávislé na použitém sázecím systému. Není-li ale k dispozici PostScriptový font s úplnou sadou českých znaků, pak je nutno provést „rozklad“ akcentovaných znaků na segmenty už na úrovni T-kódování ⟶ F-kódování, což je ovšem výsadou TEXovských instalací a virtuálních fontů.

Výsadou TEXovských instalací je rovněž ovladač `dvips`, kterému můžeme sdělit požadovaný Encoding vektor použitého fontu přímo prostřednictvím konfiguračního souboru `psfonts.map`. Je-li tam například napsáno:

```
csr10 dcr10 "XL2encoding ReEncodeFont" <xl2.enc <dcr10.pfb
```

pak se pro sazbu fontu `csr10` použije font `dcr10.pfb`, ovšem s pozměněným Encoding vektorem. Program `dvips` zavede do PostScriptového kódu font `dcr10.pfb`, ale změní mu Encoding vektor tak, jak je řečeno v souboru `xl2.enc`. To je textový soubor, kde je požadovaný Encoding vektor zapsán. Nemusíme tedy tento vektor měnit přímo v souborech `pfb` a `pfa`. Podrobněji viz [DVIPS].

Reference

[WWW] Stránka o kódování češtiny: `http://www.cuni.cz/cestina`.

[FTP] Přehledné a důkladné tabulky jedenácti kódování používaných v češtině: `ftp://math.feld.cvut.cz/pub/tkadlec/text/codestab.ps`. Dostupné též ze stránky: `http://math.feld.cvut.cz/tkadlec/others.htm`

[RB] Adobe Systems Incorporated. *PostScript Language Reference Manual.* Addison-Wesley, Reading, MA, USA, second edition, 1990.

[DVIPS] Tom Rokicki. `dvips` — *dvi to PostScript*. Manuál k programu `dvips` ve formátu `*.tex` přiložen v distribuci programu.
[EC] Milan Zamazal. Balík pro počeštění Emacsu `emacs-czech`, `http://www.fi.muni.cz/~pdm/emacs-czech.html`.
[ENCTEX] Petr Olšák. Balík encTEX rozšiřující standardní TEX o možnost manipulace s `xord/xchr` vektory. `ftp://math.feld.cvut.cz/pub/olsak/enctex/`. Petr Olšák: *EncTEX — změny konverzních tabulek TEXu*. Zpravodaj Československého sdružení uživatelů TEXu, **3** (7), 109–118 (1997).
[TBN] Petr Olšák. *TEXbook naruby*. Vyjde v nakladatelství Konvoj. V elektronické podobě plný text k dispozici na `http://math.feld.cvut.cz/olsak/tbn/`.
[TST] Petr Olšák. *Typografický systém TEX*. CSTUG 1995.

LaTeXová kuchařka/3

Zdeněk Wagner

Třetí díl LaTeXové kuchařky naváže na předchozí část. Naposledy se zastavíme u maker pro změnu velikosti písma. Poté nás čeká drobné intermezzo týkající se robustních a křehkých příkazů a pak už se můžeme věnovat vzhledu obsahu.

13. Změna řádkového prokladu

Na začátku této kapitoly mi připadá nemilá povinnost, a tou je oprava omylu z předchozí části [1]. Týká se definice makra `\@setfontsize`. První parametr je použit jinak, než bylo uvedeno. Slouží totiž k tomu, aby se aktuální velikost písma uložila do `\@currsize`. Definice vypadá přibližně takto:

```
\def\@setfontsize#1#2#3{%
  \let\@currsize #1\fontsize{#2}{#3}\selectfont}
```

Prvním parametrem je tedy makro, v jehož definici se `\@setfontsize` vyskytuje. Další dva parametry specifikují velikost písma a vzdálenost účaří. Poslední dvě hodnoty budou uloženy do vnitřních maker `\f@size` a `\f@baselineskip`. Tyto uschované hodnoty lze s výhodou použít v případech, kdy chceme makrem `\fontsize` změnit pouze jednu z nich.

Zde vidíme další důvod, proč bychom neměli používat `\fontsize` přímo. Pak totiž `\@currsize` obsahuje nesprávný příkaz a balíky, které na toto makro spoléhají, mohou způsobit katastrofu.

Zatím jsme si ale neřekli, jak skutečně funguje nastavení vzdálenosti účaří, potažmo řádkového prokladu. Musíme se podívat, jak TEX láme odstavce na řádky. Každý řádek je uložen do boxu a příslušné boxy jsou skládány pod sebe tak, aby vzdálenost účaří byla rovna hodnotě registru `\baselineskip`. Přitom se kontroluje vzdálenost horního okraje právě vkládaného boxu od spodního okraje předchozího boxu. Ta nesmí být menší než `\lineskiplimit`. Pokud by tato podmínka nebyla splněna, vložil by TEX mezi boxy vertikální mezeru velikosti `\lineskip`. Hodnoty `\lineskip` i `\lineskiplimit` můžeme předefinovat. Navíc můžeme změnit hloubku předchozího boxu, kterou TEX uložil do registru `\prevdepth`.

Příkaz pro změnu řádkového prokladu se nabízí sám. Stačí změnit hodnotu `\baselineskip`. Před takovým přístupem je ale nutno důrazně varovat! Registr `\baselineskip` je pro uživatele LATEXu tabu. Je pravda, že v jednoduchých dokumentech to funguje podle očekávání, ale často se dočkáte nemilých překvapení. Může za to právě makro `\fontsize`, které mění kromě velikosti písma i vzdálenost účaří. Stačí tedy jakákoliv změna velikosti písma a vaše nastavení `\baselineskip` je ztraceno. Lze namítat, že změny velikosti provádíme obvykle lokálně uvnitř skupiny, takže po jejím uzavření se `\baselineskip` obnoví. To je sice pravda, ale nesmíme zapomínat, že řada standardních maker v sobě obsahuje například `\normalsize`, a tím o své nastavení registru `\baselineskip` přijdeme.

V LATEXu máme tři metody, jak změnit řádkový proklad. První z nich jsme si ukázali již v předchozí části kuchařky. Stačí předefinovat `\normalsize`, `\small`, `\large` a ostatní makra pro změnu velikosti písma. V odpovídajících příkazech `\fontsize` uvedeme požadovanou vzdálenost účaří. Tuto metodu použijeme, pokud píšeme vlastní styl pro dokument určitého typu. Tak je to nastaveno i ve stylu pro sazbu tohoto Zpravodaje. Nevýhodou takového postupu je skutečnost, že řádkový proklad je definován kdesi uvnitř chráněných maker a nelze jej tudíž snadno ovlivňovat na uživatelské úrovni.

Autor se často dostává do situace, že by potřeboval v prví fázi tvorby rukopisu zvětšit řádkový proklad, aby si mezi řádky mohl vpisovat poznámky. I na to LATEX pamatuje. Musíme si ovšem uvědomit, jak funguje makro `\fontsize`. Když opomeneme příkazy pro uschování aktuálních hodnot, můžeme jeho definici zjednodušeně zapsat:

```
\def\fontsize#1#2{%
  \set@fontsize\baselinestretch{#1}{#2}}
```

Změna velikosti je makrem `\set@fontsize` pouze zaznamenána, ale provede ji až `\size@update` při provádění příkazu `\selectfont`. Za normálních okolností nedělá `\size@update` nic. Makro `\set@fontsize` uschová své tři parametry do `\f@linespread`, `\f@size` a `\f@baselineskip`. Hodnota `\f@linespread` je pak vrácena zpět do `\baselinestretch`. Potom je přede-

finováno makro `\size@update` tak, aby se vzdálenost účaří nastavila na součin `\f@linespread` a `\f@baselineskip`. Vlastní činnost je pak provedena během expanze makra `\selectfont`. Změny řádkování tedy docílíme předefinováním `\baselinestretch`, například:

```
\renewcommand\baselinestretch{1.2}
```

LaTeX 2ε nabízí navíc makro `\linespread`, které je definováno (přibližně):

```
\def\linespread#1{\set@fontsize{#1}\f@size\f@baselineskip}
```

Stejného výsledku jako výše dosáhneme nyní velmi jednoduchým povelem `\linespread{1.2}`.

Zde je nutno připojit komentář pro ty, kdo ještě používají zastaralý LaTeX 2.09. Jak patrno, nová hodnota `\baselinestretch` se projeví až po změně velikosti písma. Bohužel pouhý `\normalsize` k tomu nestačí. Toto makro totiž bylo optimalizováno a ignorovalo se, pokud již byla zvolena základní velikost. Uživatel tedy musel napřed zmenšit velikost například na `\small` a potom vrátit na `\normalsize`. Pro LaTeX 2ε stačí použít `\selectfont`. Toto makro automaticky provede změnu velikosti písma i v tom případě, že současná hodnota `\baselinestretch` neodpovídá uschované `\f@linespread`.

Třetí metodou změny řádkového prokladu je přímé použití příkazu `\fontsize`. To má opodstatnění v krátkých tiskovinách, jako jsou třeba vizitky, na titulních stránkách, nebo jako součást maker, která mění řádkový proklad pouze lokálně v malé části dokumentu. Nejbližší `\normalsize` totiž vrátí `\baselineskip` zpět na standardní hodnotu.

14. Křehká a robustní makra

Tato odbočka je nesmírně důležitá pro pochopení následujícího textu. Často se osobně i v diskusním listu setkávám s dotazy, proč LaTeX hlásí chybu v `\caption`, ale přitom je vše vytištěno správně. Zřejmě již tušíte, že je to způsobeno použitím křehkého makra. Jak ale předem poznáme křehké makro od robustního? Odpověď není těžká, ale zatím ji odložíme.

Začneme trochu zeširoka. Víme, že TeX čte vstupní proud a předává jej k dalšímu zpracování. Pokud při čtení objeví nějaký token, který se dá expandovat, provede expanzi a výsledek vloží zpět do vstupního proudu. Ten může být předmětem další expanze. Jsou však situace, kdy k expanzi nedojde. To je případ již zmíněného makra `\caption`. LaTeX jej totiž definoval jako makro s parametry. Při jeho expanzi tedy TeX nejprve parametry hledá. Předpokládejme, že nepovinný parametr nebyl uveden, takže TeX vezme text mezi složenými závorkami a uschová jej jako parametr makra. Tento text je pak vysázen a přitom dojde k plné expanzi všech použitých maker. To nečiní žádné potíže. Problém však nastává, když se parametr zapisuje do pomocného souboru, z něhož se při dalším průchodu vytvoří seznam obrázků či seznam tabulek. Zápis provádí primitiv

`\write` a ten také expanduje veškerá makra v zapisovaném textu. Některá makra však v této situaci expandovat nelze, a právě to způsobí chybu. Pokud netisknete seznam obrázků ani seznam tabulek, chyba se v dokumentu neprojeví. Robustní makro je ochráněno před takovou předčasnou expanzí. Do pomocného souboru se zapíše bez expanze a ke skutečné expanzi dojde až v okamžiku tisku.

Křehké makro musíme ochránit použitím `\protect`. Toto ochranné makro má proměnný význam podle situace, v níž se vyskytuje. Dokumentovaný zdrojový kód LaTeXu uvádí pět možností, které mohou nastat. Makro `\protect` může být definováno jako:

- `\relax` – Toto se používá při normální sazbě. Potom `\protect\foo` provede `\foo`. Dochází k plné expanzi všech maker. Uživatel nesmí `\protect` pro tuto základní situaci změnit. Řada vnitřních maker na tuto definici spoléhá. Pokud potřebujete jinou definici makra `\protect` pro běžnou sazbu, musíte stejným způsobem předefinovat i `\@typeset@protect`. Nejvhodnější je předefinovat toto pomocné makro a pro definování makra `\protect` použít `\set@typeset@protect`.
- `\string` – Používá se pro výpis na obrazovku. Makra `\protect\foo` pak vypíší `\foo`. Tuto hodnotu lze do makra `\protect` vložit příkazem `\set@display@protect`.
- `\noexpand` – To je vhodné pro zápis do souboru, který se bude dále zpracovávat (La)TeXem. Zde potřebujeme, aby za makrem byla vynechána mezera. Sekvence `\protect\foo` nyní skutečně zapíše `\foo` s mezerou.
- `\@unexpandable@protect` – Občas potřebujeme do souboru zapsat makro tak, aby bylo robustní a sneslo další zacházení. Výsledkem zápisu `\protect \foo` musí být opět `\protect\foo` následované mezerou. Již jsme uvedli, že část úkolu splní `\noexpand\foo`. Stejným způsobem potřebujeme předem zapsat `\protect`, takže kompletní sekvence příkazů bude `\noexpand \protect\noexpand\foo`. Z úsporných důvodů definuje LaTeX zkratku:
  `\def\@unexpandable@protect{\noexpand\protect\noexpand}`
  Makro se používá i při definicích prostřednictvím `\edef` a `\xdef`. Pro tyto účely máme další zkratky, `\protected@edef` a `\protected@xdef`. Pokud po `\xdef` nepotřebujeme obnovit původní `\protect` (protože definici provádíme uvnitř skupiny a obnovu tudíž zařídí TeX), můžeme použít `\unrestored@protected@xdef`. Definice najdete v souboru `ltdefns.dtx`. Zde je drobná ukázka:

```
def\protected@edef{%
  \let\@@protect\protect
  \let\protect\@unexpandable@protect
  \afterassignment\restore@protect
  \edef
}
\def\restore@protect{\let\protect\@@protect}
```

- `\@unexpandable@noexpand` – Původně bylo určeno pro odložený zápis uvnitř `\edef`. Mělo způsobit zapsání `\foo` a mezery do souboru. Definice tohoto makra byla:
  `\def\@unexpandable@noexpand{\noexpand\noexpand\noexpand}`
  Všimněte si minulého času. Toto makro totiž nebylo nikdy využito, takže již v LaTeXu definováno není.

Řekli jsme si, že `\protect` chrání křehká makra před expanzí. Víme také, že TeX v jistých situacích expanduje vše až na primitivy. Z tohoto hlediska jsou tedy všechna definovaná makra křehká. Některá makra jsou definována tak, že je lze bezpečně expandovat ve všech režimech, ale jsou případy, kdy si to dovolit nemůžeme. Předpkládejme, že `\Macro` patří do této kategorie. Zkusme nyní přidat následující definici:

```
\def\MyMacro{\protect\Macro}
```

Snadno se přesvědčíme, že `\MyMacro` bude v běžném textu dělat totéž co `\Macro` – až na drobné výjimky[1]. Navíc bude nové makro fungovat i v pohyblivých argumentech. Jeho expanzí totiž vznikne `\protect\Macro`. Naše `\MyMacro` je tedy robustní.

Při definici robustních maker musíme být pečliví. V jednom svém dokumentu jsem často potřeboval psát název programu – „Gepard“. Abych nemusel psát backslash nebo prázdné složené závorky za makrem, vytvořil jsem si `\g+`. Makro se mělo vyskytovat i v názvech kapitol a z nich se mělo přenést do obsahu, muselo tedy být robustní. Zcela mechanicky jsem pak zapsal:

```
\def\g+{\protect\pg}
\def\pg{Gepard}
```

V běžném textu vše fungovalo správně. Mezera po `\g+` nebyla spolknuta. Do pomocného souboru se však zapsalo `\protect\pg` následované mezerou. Při následujícím zpracování ale TeX tuto mezeru spolkne, takže za slovem „Gepard“ mezera chyběla. Definice měla správně vypadat:

```
\def\g{\protect\pg}
\def\pg+{Gepard}
```

Tak se definují robustní makra. Nevýhodou tohoto postupu je nutnost definice dvou maker. Řekneme uživateli, že má pro určitou činnost použít `\MyMacro` a že je toto makro robustní. Nyní si uživatel nadefinuje `\Macro` a dojde ke katastrofě. Musíme tedy ještě informovat, jaké makro uživatel předefinovat nesmí. Tuto nevýhodu obejdeme použitím definičního příkazu `\DeclareRobustCommand`. Používá se naprosto stejně jako `\newcommand`. Při definici robustního makra se vytváří vnitřní pomocné makro s tak obskurním jménem, že se jej nikomu nepodaří předefinovat – kromě největších čarodějů, ale ti vědí, co dělají.

[1] Pokud expandované `\Macro` začíná číslicí, pak `1\Macro` bude mít jiný účinek než `1\MyMacro`.

Jak tedy poznáme robustní makro? Požádáme TEX, aby vypsal jeho definici. Zkusme třeba:

```
\show\MyMacro
```

Na obrazovku i do `log` souboru se vypíše:

```
> \MyMacro=macro:
->\protect \Macro .
```

Podobný výpis dostanete pro každé robustní makro.

15. Vytváření obsahu

Řekli bychom, že vytvoření obsahu v LATEXovém dokumentu je triviální záležitostí. Stačí přece napsat `\tableofcontents`. Toto tvrzení je sice pravdivé, ale pouze částečně. Uvedeným příkazem totiž vysázíme obsah v předpřipraveném formátu. Často však potřebujeme vzhled obsahu změnit. V následujícím textu si povíme, jak lze nejrůznějších změn snadno dosáhnout.

15.1. Jak LATEX vytváří obsah

Chceme-li zasahovat do tvorby obsahu, musíme nejprve něco vědět o mechanismu, který standardní LATEXové styly pro sazbu obsahu používají. Makra `\part`, `\chapter`, `\section` a další ukládají název části či kapitoly včetně čísla stránky do pomocného souboru s příponou `.toc`. Nebudeme popisovat podrobně mechanismus, uvedeme pouze, že tento pomocný soubor obsahuje pro každou kapitolu makro `\contentsline` s příslušnými parametry.

Makro `\contentsline` má tři parametry. Prvním parametrem je typ záznamu, tedy například `chapter` nebo `section`, druhým parametrem je vlastní text příslušného názvu a třetím parametrem je číslo stránky. Číslo kapitoly je uváděno v argumentu makra `\numberline`. Ve Zpravodaji č. 3/1996 obsahuje pomocný soubor mimo jiné:

```
\contentsline {section}{\numberline {A}Úvod}{134}
\contentsline {section}{\numberline {B}Původ}{135}
\contentsline {subsection}{\numberline {1}Co je to \TeX {}}{135}
\contentsline {subsection}%%%
   {\numberline {2}Jak mám vyslovovat \uv {\TeX }}{136}
\contentsline {subsection}{\numberline {3}Co to je \MF {}}{136}
\contentsline {section}%%%
   {\numberline {C}Dokumentace a\nobreakspace {}nápověda}{145}
\contentsline {subsection}%%%
```

```
   {\numberline {18}Knihy o \TeX {}u a příbuzná literatura}{145}
\contentsline {subsection}%%%
   {\numberline {19}Kde lze najít tento dokument}{146}
```

Některé řádky jsou příliš dlouhé, proto musely být ručně zlomeny, aby se vešly do tiskového zrcadla. Místo ručního zlomu je označeno třemi procenty a odsazením druhé části zlomeného řádku.

Makro `\tableofcontents` vysází nadpis obsahu, obvykle příkazem:

```
\chapter*{\contentsname}
```

Ve stylu ARTICLE se pochopitelně použije `\section*`. Makro `\contentsname` je definováno ve všech jazykových stylech tak, aby fungovalo v angličtině, češtině, francouzštině, jakož i v jiných jazycích. Po vysazení nadpisu se zavolá makro `\@starttoc`, které načte pomocný soubor a otevře jej znovu pro zápis. Soubor `article.cls` obsahuje následující definici:

```
\newcommand\tableofcontents{%
  \section*{\contentsname
    \@mkboth{%
      \MakeUppercase\contentsname}{\MakeUppercase\contentsname}}%
  \@starttoc{toc}%
  }
```

Makro `\@mkboth` navíc vytvoří živé záhlaví. Pokud nevíte, proč je použito makro `\MakeUppercase` místo primitivu `\uppercase`, přečtěte si článek Petra Olšáka: Putování písmene ř z klávesnice na papír [3].

Makro `\@starttoc` má jeden parametr, a tím je přípona pomocného souboru. Pak tedy `\@starttoc{toc}` vysází obsah, `\@starttoc{lof}` je voláno makrem `\listoffigures` a vytiskne seznam obrázků s použitím údajů z pomocného souboru s příponou `.lof`. Z pomocného souboru se načítají příkazy `\contentsline{typ}`, které se expandují na `\l@typ`. Pokud se nám tedy nelíbí, jak se v obsahu tiskne název kapitoly, musíme předefinovat `\l@chapter`.

LaTeX zapisuje do obsahu pouze kapitoly a podkapitoly do určité úrovně. Představte si, že chcete počet úrovní v obsahu rozšířit. Pohledem do pomocného souboru zjistíte, že příslušné údaje v něm jsou, ale v obsahu se neobjeví. LaTeX má totiž určitý mechanismus, jímž se určuje, jaká nejhlubší úroveň nadpisů se v obsahu vytiskne. Přehled úrovní najdete v tabulce 1. Všimněte si, že se mění úroveň nadpisu `\part` podle třídy dokumentu. Je to tak proto, aby nadpisy `\section` a méně významné měly stejnou úroveň, čímž se další programování maker významně zjednoduší. K této tabulce se vrátíme ještě v dalších dílech.

Úroveň nadpisů, které se objeví v obsahu, ovlivníme čítačem `tocdepth`. Pokud jej nastavíme na hodnotu 2 příkazem:

```
\setcounter{tocdepth}{2}
```

Tabulka 1: Přehled úrovní standardních nadpisů

`\part` (book, report)	-1	`\part` (article)	0
`\chapter`	0	`\section`	1
`\subsection`	2	`\subsubsection`	3
`\paragraph`	4	`\subparagraph`	5

budeme v obsahu mít všechny nadpisy do úrovně 2 včetně, tedy `\part`, `\chapter`, `\section` a `\subsection`.

Ukážeme si to na příkladech. Pro jednoduchost vynecháme vlastní nadpis „Obsah“ a všechny ukázky budou odděleny od textu vodorovnými linkami. Nejprve uvedeme část obsahu Zpravodaje 3/96 (viz obsah pomocného souboru na str. 145) s čítačem `tocdepth` nastaveným na hodnotu 1.

Obsah je velmi stručný. Podrobnější verzi obsahu získáme nastavením čítače `tocdepth` na hodnotu 2.

15.2. Kde a jak je definován vzhled obsahu?

Předvedli jsme si způsob, jak rozhodneme, které nadpisy budou do obsahu zahrnuty. Nyní se již budeme věnovat změnám jeho vzhledu. Nejprve nás bude ovšem zajímat, odkud LATEX příslušné definice bere.

Pozorný čtenář si jistě pamatuje, že se do pomocného souboru zapisují příkazy `\contentsline{typ}`, které se během tisku obsahu expandují na `\l@typ`. Makro `\section` tedy zapíše do pomocného souboru `\contentsline{section}`, a to se bude později expandovat na `\l@section`. Posledně jmenované makro společně s dalšími makry pro tisk obsahu je uvedeno v souboru definujícím třídu dokumentu. Tak například soubor `article.cls` obsahuje (`\l@part` úmyslně vynecháme):

```
\newcommand*\l@section[2]{%
  \ifnum \c@tocdepth >\z@
    \addpenalty\@secpenalty
    \addvspace{1.0em \@plus\p@}%
    \setlength\@tempdima{1.5em}%
    \begingroup
      \parindent \z@ \rightskip \@pnumwidth
      \parfillskip -\@pnumwidth
      \leavevmode \bfseries
      \advance\leftskip\@tempdima
      \hskip -\leftskip
      #1\nobreak\hfil \nobreak\hb@xt@\@pnumwidth{\hss #2}\par
    \endgroup
  \fi}
\newcommand*\l@subsection{\@dottedtocline{2}{1.5em}{2.3em}}
\newcommand*\l@subsubsection{\@dottedtocline{3}{3.8em}{3.2em}}
\newcommand*\l@paragraph{\@dottedtocline{4}{7.0em}{4.1em}}
\newcommand*\l@subparagraph{\@dottedtocline{5}{10em}{5em}}
\newcommand*\l@figure{\@dottedtocline{1}{1.5em}{2.3em}}
\let\l@table\l@figure
```

Obvykle požadujeme tečkovanou linku mezi názvem a číslem strany. Pro usnadnění používáme `\@dottedtocline`, o němž se ještě za chvíli zmíníme. Jak bylo ukázáno v předchozích příkladech, `\section` se v obsahu vysází jiným písmem, bez tečkované linky a s vertikální mezerou. Proto je také definice `\l@section` odlišná. Na první pohled vypadá složitě, ale věřte, že to ve skutečnosti vůbec složité není. Význam budeme odhalovat postupně s výkladem, jak lze vzhled změnit k obrazu svému.

Začneme sdělením, že všechna makra `\l@typ` požadují dva parametry. Prvním parametrem je název (pod)kapitoly, druhým je číslo strany. Tyto informace se tisknou ke konci definice `\l@section`. Připomeneme, že `\hb@xt@` je zkratka za `\hbox to` a šetří několik tokenů (to je významné zejména při generování formátu). Celá konstrukce

```
\hb@xt@\@pnumwidth{\hss #2}
```

je funkčně ekvivalentní známému (alespoň pro LaTeXisty) zápisu

```
\makebox{\@pnumwidth}[r]{#2}
```

Makro `\makebox` ale prochází dosti složitou expanzí, a proto je výrazně méně efektivní než použité `\hb@xt@`.

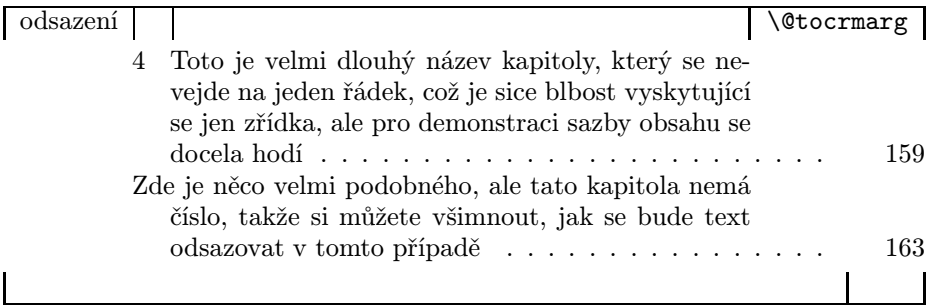
Pozorný čtenář má již delší dobu na jazyku jedovatou poznámku. Makro `\l@section` má sice dva parametry, o čemž svědčí dvojka v hranatých závorkách, ale `\l@subsection` ani další makra žádné parametry nemají! Pozorný čtenář má pravdu, ale pouze částečnou. Trik spočívá v tom, že makro `\@dottedtocline` požaduje pět parametrů, ale v definicích jsou použity pouze tři. `\l@subsection` své dva parametry nespotřebuje a nechá je ve vstupním proudu netknuté. Po expanzi si je vezme `\@dottedtocline` jako čtvrtý a pátý parametr.

Poslední dva parametry makra `\@dottedtocline` již tedy známe. Nyní vysvětlíme první tři. Z pohledu na definice a kontrolou tabulky 1 asi tušíte, že prvním parametrem je úroveň nadpisu. Ta se porovnává s hodnotou čítače `tocdepth`, čímž se zjistí, zda se má příslušná položka v obsahu vytisknout. Makro `\l@section` provádí tento test na začátku primitivem `\ifnum`. Druhý parametr určuje odsazení od levého okraje tiskového zrcadla a třetí parametr vyhrazuje prostor pro číslo (pod)kapitoly. Stejná hodnota se použije též pro odsazení, pokud je název víceřádkový. Tehdy musíme také definovat vzdálenost, kde se mají řádky zlomit. Ta je uložena v makru `\@tocrmarg`. Podobně do makra `\@pnumwidth` vložíme požadovanou šířku boxu pro stránkovou číslici. Příkladem takových definic může být:

```
\renewcommand*\@tocrmarg{23mm}%
\renewcommand*\@pnumwidth{10mm}%
\renewcommand*\l@section{\@dottedtocline{1}{17mm}{5mm}}
```

Všechny rozměry jsme uvedli v milimetrech, abyste si je mohli odměřit obyčejným pravítkem. Příklad výstupu je uveden na obrázku 1.

Obrázek 1: Demonstrace parametrů makra `\@dottedtocline`



Číslice 4 je odsazena od textu, protože je uzavřena v `\numberline{4}`. Toto makro má následující definici:

```
\def\numberline#1{\hb@xt@\@tempdima{#1\hfil}}
```

Hodnota rozměru `\@tempdima` se vezme z třetího parametru makra `\@dottedtocline`. V případě makra `\l@section` je to však trochu jinak. Podíváme se na jeho definici znovu.

Makro začíná svoji činnost porovnáním hodnoty čítače `tocdepth` s nulou. Je-li hodnota čítače větší, má být nadpis v obsahu vytištěn, v opačném případě se přeskočí vše až k primitivu `\fi` včetně. Primitiv `\ifnum`, stejně jako dále se vyskytující `\begingroup` a `\endgroup`, již nejsou čistě LaTeXové konstrukce a pro jejich pochopení si musíte pročíst alespoň jednu z knih [4] nebo [5]. Pokud je tedy podmínka splněna, přidá se záporná penalizace, která informuje o tom, že zde je vhodné místo pro stránkový zlom. Dále následuje vertikální mezera. Ta bude samozřejmě vypuštěna, pokud by se dostala na začátek stránky. Zařídí to automaticky algoritmy pro stránkový zlom. Potom vložíme vhodnou velikost prostoru pro číslo kapitoly do proměnné `\@tempdima`. Uvnitř skupiny nastavíme parametry odstavce a vše vytiskneme tučným písmem.

Teď již umíme dělat jednoduché zásahy do vzhledu obsahu. Podívejme se znovu na náš příklad. Názvy kapitol nám tam připadají příliš tučné, chybí u nich tečkovaná linka, zatímco u podkapitol je zbytečně velký prostor pro číslo podkapitoly. Změníme tedy definice následujícím způsobem:

```
\renewcommand*\l@section{\@dottedtocline{1}{0mm}{1.5em}}
\renewcommand*\l@subsection{\@dottedtocline{2}{1.5em}{1.5em}}
```

Obsah pak bude vypadat takto:

Při sazbě knihy s velmi hlubokým členěním je někdy odsazování na závadu. Stačí ovšem jednoduchá předefinice, např.:

```
\renewcommand*\l@section[2]{\@dottedtocline{1}{0mm}{1.5em}%
                            {\bfseries #1}{\bfseries #2}}
\renewcommand*\l@subsection{\@dottedtocline{2}{0mm}{1.5em}}
```

Výsledkem pak bude:

Také můžeme zcela opustit makro `\@dottedtocline` a vytvořit nové definice třeba takto:

```
\renewcommand*\l@section[2]{\@tempdima 1.5em
          \noindent\hb@xt@\@pnumwidth
          {\hss\bfseries #2:}\enspace \textbf{#1}\par}
\renewcommand*\l@subsection[2]{\@tempdima 1.5em
          \noindent\hb@xt@\@pnumwidth
          {\hss #2:}\enspace #1\par}
```

Zde vidíte výsledek:

15.3. Jak se dostanou informace do pomocného souboru?

Můj školitel kdysi říkával, že život není žádná humanita. Platí to beze zbytku i v LATEXu. Ukázali jsme si, že záznamy z pomocného souboru lze využít k vysázení obsahu v mnoha různých podobách. V praxi se nám ale může stát, že potřebná informace v pomocném souboru není. O její uložení do pomocného souboru se musíme postarat sami. Proto se nejprve podíváme, jakým způsobem provádí zápisy do pomocného souboru sám LATEX.

Nebudeme se zabývat podrobným rozborem všech maker, protože to uživatel k ničemu nepotřebuje. Nebudou nás zajímat všechny cestičky, jimiž se budou

písmenka ubírat, než dojdou na správné místo správného souboru. Chceme pouze najít vhodné makro, které splní námi požadovaný úkol, a bude stačit, když mu porozumíme na uživatelské úrovni.

Pro zápis informací, které se mají objevit v obsahu, máme dvě makra: `\addcontentsline` a `\addtocontents`. Každé z nich má své použití.

Makro `\addtocontents` tvoří základ pro zápis informací do obsahu. Obvykle je automaticky voláno makry jako `\section` a `\caption`. Má tři parametry: typ souboru (uvádí se přípona, nejčastěji `toc` = table of contents, `lof` = list of figures, `lot` = list of tables), typ záznamu (chapter, section, figure, apod.) a vlastní text nadpisu. Je zřejmé, že křehká makra použitá v nadpisu vyžadují `\protect`. Jako příklad si uvedeme makro `\chapter`, které během své expanze použije:

```
\addcontentsline{toc}{chapter}%
                  {\protect\numberline{\thechapter}nadpis}
```

Zatím se nebudeme starat, jak se dostane *nadpis* z parametru makra `\chapter` do třetího parametru `\addcontentsline`. To si vysvětlíme v jednom z dalších pokračování, až se budeme zamýšlet nad vzhledem nadpisů. Všimneme si hlavně použití makra `\thechapter`, které obsahuje tisknutelnou podobu čítače `chapter`. V zásadě lze říci, že každý typ nadpisu má přiřazen stejnojmenný čítač, takže pro `\subsection` bychom použili `\thesubsection`, pro nadpis obrázku (figure) použijeme `\thefigure` apod.

Existuje typická situace, kdy použijeme `\addcontentsline` sami. Chceme například v knize nadpis jedné kapitoly vytisknout bez čísla. K tomuto účelu slouží varianta nadpisového makra s hvězdičkou. Pak se nám ale nadpis nedostane do obsahu. Zde tedy musíme připsat `\addcontentsline`. Ukážeme si to na příkladu kapitoly „Úvod“, přičemž navíc požadujeme, aby text byl v obsahu odsazen o místo vyhrazené pro číslo kapitoly. V dokumentu proto uvedeme:

```
\chapter*{Úvod}
\addcontentsline{toc}{chapter}{\protect\numberline{}Úvod}
```

Možná se ptáte, jak se do obsahu dostane číslo stránky. Při své expanzi jej makro `\addcontentsline` doplní samo. Proto musí být `\addcontentsline` uvedeno *bezprostředně po* vytištění odpovídajícího nadpisu, protože jinak by číslo stránky v obsahu mohlo být nesprávné.

Makro `\addtocontents` má jen dva parametry: typ souboru a text. Používá se ke vkládání rozličných informací nesouvisejících s nadpisy; proto se také nedoplňuje číslo stránky. Můžeme například vložit vertikální mezeru příkazem

```
\addtocontents{toc}{\protect\vspace{1ex}}
```

Tímto způsobem také můžeme měnit definice maker. Makro `\@starttoc` totiž načítá pomocný soubor uvnitř skupiny. Pokud vložíme do druhého

parametru makra `\addtocontents` definice maker, budou tyto definice platit pouze do okamžiku ukončení tisku obsahu.

Podle návodu v této podkapitole dokážeme sestavit další typy obsahu. Představme si, že v dokumentu máme velké množství definic. Budeme je uvozovat makrem `\definice`, jehož parametrem bude název definice. Tento název chceme vytisknout v seznamu definic. Všechny definice budeme číslovat s použitím čítače `definice` a pro vytištění seznamu si vymyslíme makro `\seznamdefinic`. Příslušná makra mohou vypadat přibližně takto:

```
\newcommand*\definice[1]{%
  \refstepcounter{definice}% inkrementace čítače
  % Zde budou příkazy pro tisk čísla a názvu definice
  \addcontentsline{def}{definice}%
                  {\protect\numberline{\thedefinice}#1}}
\newcommand*\l@definice{\@dottedtocline{0}{0em}{1.3em}}
\newcommand*\seznamdefinic{\chapter*{Seznam definic}
   \@starttoc{def}}
```

15.4. Poněkud špinavé triky

Celý mechanismus vytváření obsahu jsme si již popsali z uživatelské stránky. Nic nám nebrání ve zkoumání, co vše LaTeX snese. Jako úvod pro další experimenty lze uvést několik příkladů. Nebudou to ukázky akademické. Nejrůznější formy vzhledu obsahu budou demonstrovány na tom, co přinesl život, tedy přesněji zákazníci, kteří měli zcela vyhraněné požadavky.

Začneme Zpravodajem, který právě čtete. Víte, že existuje styl `csbul.sty`[2], který se pro sazbu používá. Víte také, že `\section` se do obsahu nepřenese, a použitá třída ARTICLE nezná `\chapter`. Jak je to zde uděláno? Vše řeší následující makra:

```
\newif\ifautkn@wn
\renewcommand\maketitle[1][\z@]{%
  % příkazy pro tisk nadpisu
  \ifautkn@wn % true, pokud je za \begin{clanek} uveden autor
    \addcontentsline{toc}{clanek}{\protect\cl@nek{\auth@r}{\t@tle}}%
  \else
    \addcontentsline{toc}{clanek}{\t@tle}%
  \fi
  % další příkazy
  }
\def\cl@nek#1#2{#1: #2}
\def\l@clanek#1{% příkazy pro výstup do HTML
  \@dottedtocline{-2}{\z@}{2em}{#1}}
```

[2] `ftp://ftp.icpf.cas.cz/wagner/cstex/bulletin/csbuldoc.zip`

Všimněte si, že jsme vůbec nemuseli měnit definici `\tableofcontents`. Ve skutečném stylu `csbul.sty` ji sice měníme, ale z jiného důvodu. Chceme totiž současně vytisknout obsah ve formátu HTML pro zveřejnění na WWW. Proto i `\l@clanek` má několik dalších příkazů navíc, které používají hodnotu parametru. Pro tvorbu tištěného obsahu jsou však výše uvedená makra zcela postačující.

Složitější příklad představuje německý překlad knihy Pověsti českých hradů [6]. K některému hradu je uvedena pouze jedna pověst, u jiných je pověstí více. Vždy však začínáme jménem příslušného hradu. Pokud má pověst vlastní název, uvádíme jméno hradu makrem `\hrad`. V obsahu se číslo stránky v tomto případě neuvádí, protože souhlasí se stránkou, kde začíná první pověst. Pokud se ale ke hradu váže jediná pověst, která nemá vlastní název, uvedeme jméno hradu makrem `\shrad` a do obsahu musíme číslo stránky vytisknout. Pro název pověsti vždy použijeme `\section`. Navíc nakladatel vyžadoval netradiční vzhled, kdy položky obsahu jsou centrovány a stránková číslice je oddělena lomítkem. Vše zařídíme následujícími definicemi. (Makra `\@afterindenttrue`, `\@afterindentfalse` a `\@afterheading` vysvětlíme v jednom z dalších pokračování. Spokojíme se s tím, že jejich úkolem je zajištění, aby se nezlomila stránka hned pod názvem kapitoly nebo za prvním řádkem prvního odstavce.)

```
\def\hrad#1{% příkazy pro tisk nadpisu
    \addcontentsline{toc}{hrad}{#1}
    \@afterindentfalse \@afterheading}
\def\shrad#1{% příkazy pro tisk nadpisu
    \addcontentsline{toc}{shrad}{#1}
    \@afterindentfalse \@afterheading}
\def\section#1{% příkazy pro tisk nadpisu
    \addcontentsline{toc}{section}{#1}
    \@afterindentfalse \@afterheading}
\def\l@hrad#1#2{\penalty-50\vskip 1ex{\centering\textbf{#1}\par}\nobreak}
\def\l@shrad#1#2{\penalty-50\vskip 1ex{\centering\textbf{#1\nobreak
    \hskip.8em/\nobreak\hskip.8em#2}\par}\nobreak}
\def\l@section#1#2{{\centering#1\nobreak
    \hskip.8em/\nobreak\hskip.8em#2\par}}
```

Abychom šetřili místem, vypíšeme si jen část pomocného souboru.

```
\contentsline {section}{Vorwort}{7}
\contentsline {hrad}{Libušín}{9}
\contentsline {section}{Gericht und Wahl}{9}
\contentsline {hrad}{Vyšehrad}{18}
\contentsline {section}{Die Burg über der Moldau}{18}
\contentsline {section}{Der gro\ss{}e Sprung}{26}
\contentsline {hrad}{Děvín}{33}
\contentsline {section}{Kampf und Verrat}{33}
```

```
\contentsline {hrad}{Karlštejn}{44}
\contentsline {section}{Die Karlštejner Teufel}{44}
\contentsline {section}{Von einer List}{48}
\contentsline {section}{Die treue Freundschaft}{52}
\contentsline {shrad}{Hluboká}{108}
\contentsline {hrad}{Orlík}{118}
\contentsline {section}{Das Adlernest}{118}
\contentsline {shrad}{Radyně, Kašperk}{120}
\contentsline {shrad}{Přimda}{124}
\contentsline {shrad}{Bezděz, Blatná}{132}
\contentsline {hrad}{Kost}{138}
\contentsline {section}{Die knochenharte Burg}{138}
```

V ukázce si můžete prohlédnout, jak to dopadne po vytištění.

Nyní si předvedeme poněkud složitější příklad. Výsledek je stručný, proto uvidíte celý obsah knihy Mahájánské texty Heleny Petrovny Blavacké [7]. Požadavky na vzhled obsahu byly upřesňovány během sazby, takže mezi makry

zůstaly jakési relikty. Úmyslně je ponecháváme i v tomto příkladu. Napřed se ale na makra podíváme.

```
\def\chapter#1{\clearpage\thispagestyle{plain}
  \refstepcounter{chapter}
  \addcontentsline{toc}{chapter}{\protect\numberline{\Roman{chapter}}#1}
  % Další příkazy...
  }
\def\chap#1{\clearpage\thispagestyle{plain}
  \setcounter{section}{0}
  \addcontentsline{toc}{chapter}{#1}
  % Další příkazy...
  }
\def\l@chapter{\@dottedtocline{1}{\z@}{\z@}}
\def\l@section{\@dottedtocline{2}{\tib@dim}{\Zdim}}
\def\numberline#1{{\scshape Zlomek #1.} }
\def\ln{\protect\p@ln}
\def\p@ln{\par}
\let\cln\relax
\def\toc@cln{\let\footnote\@gobble \let\p@ln\space \let\cln\\}
\AtBeginDocument{\addtocontents{toc}{\protect\toc@cln}}
\setcounter{tocdepth}{1}
```

Všimněte si, že v knize jsou použita dvě různá makra pro nadpis kapitol. Standardní `\chapter` čísluje kapitoly a zapisuje informace pro obsah. Varianta `\chap` se liší pouze tím, že kapitoly nejsou číslovány. Mohli bychom sice použít hvězdičkovou verzi `\chapter*`, ale pak bychom se o zápis do obsahu museli starat sami v dokumentu. Takto je situace uživatelsky příjemnější.

Definice maker `\l@chapter` a `\l@section` není nijak zvláštní. Za zmínku snad stojí jen použití rozměrů, jejichž hodnoty se měly vypočítat podle nějakého vzorce. Minulý čas v této větě osvětlíme za okamžik. Následuje definice makra `\numberline`. Ukazuje nové možnosti, ale definice sama je triviální.

Některé názvy mohou být dlouhé, takže se nevejdou na řádek. Ponechat rozdělení automatu není šťastné řešení, neboť zejména název má vypadat graficky příjemně a má se snadno číst. Přitom název obvykle sázíme jiným písmem (přinejmenším jinou velikostí) než obsah. Zlom řádku je proto velmi často jiný ve vlastním nadpisu a v obsahu. Nadefinujeme si tedy robustní makro `\ln`, které zlomí řádek v nadpisu, ale v obsahu vloží pouze mezeru. Podobné makro `\cln` zlomí řádek v obsahu, ale při sazbě nadpisu se chová jako `\relax`, tj. nedělá nic. Protože `\relax` je primitiv, nedochází při zápisu do pomocného souboru k jeho expanzi, a tudíž `\cln` nepotřebuje `\protect`. Při tisku obsahu musíme definice vyměnit a navíc potřebujeme zrušit tisk poznámek pod čarou. To zařídí makro `\toc@cln`, které prostřednictvím `\AtBeginDocument` zapíšeme do pomocného souboru na začátek obsahu (v této knize je obsah vytištěn na konci).

Poslední příkaz demonstruje, že jsme se rozhodli netisknout v obsahu názvy `\section`. Nakonec bylo dokonce předefinováno i makro `\section` tak, že se v něm `\addcontentsline` vůbec nevyskytuje. Makro `\l@section` je tedy relikt, jenž při zpracování knihy není použit. Současně chybí algoritmus na výpočet rozměrů `\tib@dim` a `\Zdim`. Tyto relikty zůstaly ve stylovém souboru pro případ, že by nakladatel změnil rozhodnutí.

Podívejme se nyní na výpis pomocného souboru. Při zpracování knihy došlo k expanzi českých znaků na TEXové sekvence, takže záznamy nejsou zrovna čitelné.

```
\toc@cln
\contentsline {chapter}{\'UVOD KOMENT\'ATORA}{5}
\contentsline {chapter}{P\v REDMLUVA\p@ln HELENY PETROVNY BLAVACK\'E}{8}
\contentsline {chapter}{\numberline {\uppercase {i}}HLAS TICHA}{11}
\contentsline {chapter}{\numberline {\uppercase {ii}}DV\v E STEZKY}{51}
\contentsline {chapter}{\numberline {\uppercase {iii}}SEDM BRAN}{92}
\contentsline {chapter}{POZN\'AMKY Heleny Petrovny
   Blavack\'e\footnote {Upozorn\v en\'i vydavatele: Pozn\'amky H.\,P.\,B.
   odr\'a\v zej\'i jej\'i vlastn\'i n\'azory; p\v r\'ipadn\'e omyly nebo
   nep\v resnosti byly v\nobreakspace {}textu ponech\'any tak,
   jak je autorka formulovala.}}{150}
\contentsline {chapter}{VYSV\v ETLIVKY}{165}
```

Výsledný obsah ovšem čitelný je, neboť LaTeX 2ε umí správně složit TEXové sekvence na znaky $\mathcal{CS}$-fontů (včetně `\'i`).

Ještě větší oříšek představovaly dva svazky Tajemství Tibetu [8], [9]. Bylo nutno se přizpůsobit anglickému vydání [10]. První svazek má sedm dílů, druhý svazek má dva díly. Anglická verze je typograficky nejednotná, neboť v každém dílu je členění kapitol odlišné a jiným způsobem se uvádějí nadpisy v obsahu. Přesto mají oba svazky mnoho společného, a bylo tudíž vhodné vyřešit vše společným stylovým souborem. Zde uvedeme pouze makra, která se týkají tvorby obsahu. Přesto jich bude poměrně mnoho.

```
% Counters

\def\T@fem#1{\ifcase#1 \or PRVNÍ\or DRUHÁ\or TŘETÍ\or ČTVRTÁ\or
  PÁTÁ\or ŠESTÁ\or SEDMÁ\or OSMÁ\else \@latexerr{Too large number}\fi}
\def\T@masc#1{\ifcase#1 \or PRVNÍ\or DRUHÝ\or TŘETÍ\or ČTVRTÝ\or
  PÁTÝ\or ŠESTÝ\or SEDMÝ\or OSMÝ\else \@latexerr{Too large number}\fi}

% Parts

\def\part{\clearpage\null\thispagestyle{plain}\secdef\@part\@spart}
\def\@part[#1]#2{\par
    \refstepcounter{part}
    \addcontentsline{toc}{part}{\protect\part@line{\thepart}{#1}}
    \T@switch
    % Další příkazy...
    }

% Switches

\def\B@switch{\setcounter{secnumdepth}{0}}
\def\R@switch{%
  \ifcase\c@part \or                                          %1
    \setcounter{secnumdepth}{2}%
    \def\thesection{\Roman{section}.}%
    \addtocontents{toc}{\protect\part@I}%
    \let\sec@wpul\textsc
  \or                                                         %2
    \setcounter{secnumdepth}{4}%
    \def\thesection{ČÁST~\Roman{section}.}%
    \def\thesubsection{}%
    \def\thesubsubsection{\arabic{subsubsection}.}%
    \addtocontents{toc}{\protect\part@II}%
    \let\sec@wpul\textsc
  \or                                                         %3
    \setcounter{secnumdepth}{3}%
    \def\thesection{Část~\arabic{section}:}%
    \def\thesubsection{\arabic{subsection}.}%
    \addtocontents{toc}{\protect\part@III}%
    \let\sec@wpul\wpul
  \or                                                         %4
    \setcounter{secnumdepth}{3}%
    \def\thesection{}%
    \def\thesubsection{\arabic{subsection}.}%
    \addtocontents{toc}{\protect\part@IV}%
```

```
    \let\sec@wpul\wpul
  \or                                                   %5
    \setcounter{secnumdepth}{3}%
    \def\thesection{Část~\arabic{section}:}%
    \def\thesubsection{\arabic{subsection}.}%
    \addtocontents{toc}{\protect\part@III}%
    \let\sec@wpul\textsc
  \or                                                   %6
    \setcounter{secnumdepth}{3}%
    \def\thesection{Část~\arabic{section}:}%
    \def\thesubsection{\arabic{subsection}.}%
    \let\sec@wpul\textsc
  \or                                                   %7
    \setcounter{secnumdepth}{3}%
    \def\thesection{Část~\arabic{section}:}%
    \def\thesubsection{\arabic{subsection}.}%
    \let\sec@wpul\textsc
  \fi
}

% Contents

\def\tableofcontents{\part*{\contentsname}\@starttoc{toc}\par
  \addtocontents{toc}{\protect\toc@cln}}
\def\l@part#1#2{\addpenalty{-\@highpenalty}%
  \addvspace{2.25ex plus\p@}%
  \begingroup
    \parindent\z@ \rightskip\@pnumwidth \parfillskip\z@ plus 1fill
    \interlinepenalty\@M
    \leavevmode\bfseries#1\par\nobreak
  \endgroup}
\def\part@line#1#2{\partname~#1\\#2}
\def\partline#1{\clearpage
  \addtocontents{toc}{\protect\l@part{#1}{0}}}
\def\l@chapter{\@dottedtocline{1}{\z@}{\z@}}
\def\l@section{\@dottedtocline{2}{\tib@dim}{\Zdim}}
\def\l@subsection{\@dottedtocline{3}{3.7em}{1em}}
\def\l@subsubsection{\@dottedtocline{4}{4.7em}{1em}}
\def\part@I{\settowidth\Zdim{XXVIII. }\let\numberline\r@numberline}
\def\part@II{\global\Zdim=4.7em \let\numberline\l@numberline}
\def\part@III{\global\Zdim=3.7em\relax}
\def\part@IV{\global\Zdim=\z@}
\newdimen\Zdim
\def\l@numberline#1{\hbox to\@tempdima{#1\hfil}}
```

```
\def\r@numberline#1{\hbox to\@tempdima{\hfil#1\ }}
\let\cln\relax
\def\toc@cln{\let\cln\\}
\setcounter{tocdepth}{4}
\newdimen\tib@dim

% Options

\DeclareOption{Ruzenec}{%
  \def\partname{KNIHA}
  \def\thepart{\T@fem\c@part}
  \let\T@switch\R@switch
  \tib@dim=\z@
  \let\sec@wpul\wpul
}
\DeclareOption{Bardo}{%
  \def\partname{DÍL}
  \def\thepart{\T@masc\c@part}
  \let\T@switch\B@switch
  \tib@dim=2em
  \let\sec@wpul\textsc
}
\ProcessOptions
```

Nejprve se zastavíme u čítačů. Později budeme požadovat tisk hodnoty slovně, a to někdy v mužském, jindy v ženském rodě. Proto jsme si vytvořili makra `\T@masc` a `\T@fem`. Hodnota smí být pouze v rozmezí 1–8, pokud chceme tisknout větší hodnotu, udělali jsme určitě někde chybu (takové číslo totiž v podkladech od nakladatele není).

V makru `\part` si povšimněte hlavně makra `\T@switch`. To provádí specifické přepínání, k němuž se ještě dostaneme. `\B@switch` není zajímavý, ale `\R@switch` se větví podle čísla dílu a v každé větvi se zapíše do pomocného souboru přepínací makro.

Trochu jsme změnili i definici `\tableofcontents`. Makra `\l@typ` jsou již důvěrně známá. Za povšimnutí stojí pouze skutečnost, že rozměry požadované v `\@dottedtocline` budou vypočteny později, a to v přepínacích makrech.

Nejzajímavější je část, kde definujeme volitelné parametry stylového souboru. Zde se nastaví některé rozměry a zejména se definuje chování makra `\part`.

Opět si ukážeme část pomocného souboru prvního svazku.

```
\toc@cln
\contentsline {chapter}{PŘEDMLUVA}{13}
\contentsline {part}{\part@line {PRVNÍ}{RŮŽENEC
   SKVOSTNÝCH DRAHOKAMŮ}}{29}
```

```
\part@I
\contentsline {chapter}{PŘEDMLUVA}{29}
\contentsline {chapter}{VZÝVÁNÍ A\nobreakspace {}ÚVOD}{32}
\contentsline {chapter}{DVACET OSM ZÁKLADNÍCH PŘÍKAZŮ JÓGY}{33}
\contentsline {section}{\numberline {\uppercase {i}.}Deset
   příčin bolestí}{33}
\contentsline {section}{\numberline {\uppercase {ii}.}Deset podmínek}{37}
\contentsline {part}{\part@line {DRUHÁ}{STEZKA K\nobreakspace {}NIRVÁNĚ:
   JÓGA VELKÉHO \cln SYMBOLU}}{141}
\part@II
\contentsline {chapter}{PŘEDMLUVA}{141}
\contentsline {chapter}{VZÝVÁNÍ A\nobreakspace {}ÚVOD}{144}
\contentsline {section}{\numberline
   {ČÁST\nobreakspace {}\uppercase {i}.}PŘÍPRAVNÉ NAUKY}{145}
\contentsline {section}{\numberline
   {ČÁST\nobreakspace {}\uppercase {ii}.}PODSTATNÝ OBSAH}{153}
\contentsline {subsection}{\numberline {}Obvyklá cvičení}{153}
\contentsline {subsubsection}{\numberline {1.}Jóga jednobodovosti}{153}
\contentsline {subsubsection}{\numberline {2.}Jóga nestvořeného}{179}
\contentsline {part}{\part@line {TŘETÍ}{STEZKA VĚDĚNÍ:
   JÓGA ŠESTI NAUK}}{212}
\part@III
\contentsline {chapter}{PŘEDMLUVA}{212}
\contentsline {chapter}{VZÝVÁNÍ A\nobreakspace {}ÚVOD}{215}
\contentsline {chapter}{I. NAUKA O\nobreakspace {}PSYCHICKÉM TEPLE}{217}
\contentsline {section}{\numberline
   {Část\nobreakspace {}1:}Pět přípravných cvičení}{217}
\contentsline {subsection}{\numberline {1.}Vidění
   těla jako prázdnoty}{217}
\contentsline {subsection}{\numberline {2.}Vidění
   psychického nervového systému jako prázdnoty}{218}
\contentsline {subsection}{\numberline {3.}Vidění ochranného kruhu}{221}
```

Všimněte si, jak přepínání významu makra `\numberline` změnilo vzhled jednotlivých částí obsahu.

Další příklad představuje obsah knihy Milaräpa, velký tibetský jógin [11]. V této knize každá kapitola začíná krátkým shrnutím, které chceme přenést do obsahu. Nejprve se znovu podíváme na makra, jimiž toho bylo dosaženo.

```
% Counters

\def\T@masc#1{\ifcase#1 \or PRVNÍ\or DRUHÝ\or TŘETÍ\or ČTVRTÝ\or
  PÁTÝ\or ŠESTÝ\or SEDMÝ\or OSMÝ\else \@latexerr{too large number}\fi}

% Chapters

\newtoks\chap@toc
\def\chapter{\ifhmode \par \fi \@ifstar{\s@chapter}{\n@chapter}}
\def\s@chapter#1{\def\chap@head{#1}\chap@toc{#1}\zw@chapter}
\def\n@chapter#1{%
  \refstepcounter{chapter}
  \def\chap@head{KAPITOLA \thechapter\\#1}
  \chap@toc{\protect\numberline{\thechapter}#1}\zw@chapter}
\newcommand\zw@chapter[1][\relax]{%
  \ifnewpage \clearpage \fi \newpagetrue
  \def\zw@{#1}\def\zw@next{\relax}
  \ifx\zw@\zw@next
```

```
    \let\zw@\relax
    \edef\zw@next{\noexpand\addcontentsline
                  {toc}{chapter}{\the\chap@toc}}
  \else
    \edef\zw@next{\noexpand\addcontentsline{toc}{chapter}%
      {\noexpand\protect\noexpand\cpt@entry{\the\chap@toc}{#1}}}
  \fi
  \zw@next
  % Další příkazy...
  \@afterindenttrue\@afterheading}

% Contents

\def\tableofcontents{\cleardoublepage {\toc@cln
  \centerline{\Large\textbf{OBSAH}}\vskip2\baselineskip
  \@starttoc{toc}\par}}
\def\l@part#1#2{\penalty-100\addvspace{2\baselineskip}
  \begin{center}\Large \interlinepenalty\@M
    \def\numberline##1{DÍL \T@masc ##1\\}#1
  \end{center}\nobreak}
\def\l@chapter#1#2{\addvspace{1ex plus 1ex minus 2pt}
  \begingroup
    \def\numberline##1{KAPITOLA~##1: }%
    \leftskip\parindent \rightskip\@tocrmarg
    \parfillskip -\rightskip
    \interlinepenalty\@M
    \noindent \null \hskip -\leftskip #1\nobreak
    \leaders\hbox{$\m@th \mkern\@dotsep mu
            \hbox{.}\mkern\@dotsep mu$}\hfill
    \nobreak \hb@xt@\@pnumwidth{\hfil
             \normalfont\normalcolor#2}\par
  \endgroup}
\def\cpt@entry#1{#1\\}
\def\l@section{\@dottedtocline{1}{\z@}{\Zdim}}
\def\numberline#1{\hb@xt@\@tempdima{\hss#1.\hskip.5em}}
\def\l@subsection{\@dottedtocline{2}{\parindent}{\Zdim}}
\newdimen\Zdim \Zdim=3em
\def\nl{\protect\p@ln}
\def\p@ln{\par}
\let\cln\relax
\def\ccolon{\protect\c@colon}
\def\c@colon{\par}
\def\toc@cln{\let\footnote\@gobble \let\p@ln\empty
             \def\c@colon{: }\let\cln\\}
```

```
\setcounter{tocdepth}{2}
\setcounter{secnumdepth}{1}
\newdimen\tib@dim
```

Definici čítače již nebudeme komentovat. Všimneme si však definice `\chapter`. Makro `\@ifstar` přepíná mezi dvěma dalšími makry podle toho, zda za `\chapter` byla hvězdička. Ta připraví informaci do registru `\chap@toc` buď s číslem nebo bez čísla kapitoly a zavolají `\zw@chapter`. Posledně jmenované makro má nepovinný parametr, který obsahuje ono zmíněné shrnutí. Primitivem `\ifx` testujeme přítomnost nepovinného parametru a podle okolností nadefinujeme vhodně `\zw@next`. Při definici potřebujeme expandovat token registr `\chap@toc`, proto jsme makro definovali pomocí `\edef`. Nesmíme ještě expandovat `\addcontentsline`, proto před něj připíšeme `\noexpand`. Nesmíme expandovat ani `\cpt@entry` a před ním stojící `\protect`, proto ta záhadná skupina primitivů `\noexpand`.

Makro `\tableofcontents` jsme mírně upravili a makra `\l@typ` již známe. Za povšimnutí stojí `\l@part` a `\l@chapter`, která si lokálně předefinovávají `\numberline`, a `\chap@toc` používané pro tisk souhrnu kapitoly. Makro `\toc@cln` i jeho „přátele“ již také známe z předchozích příkladů. Zde se navíc musíme postarat o dvojtečku.

Můžeme tedy přistoupit k zobrazení části pomocného souboru:

```
\contentsline {chapter}{PŘEDMLUVA KOMENTÁTORA}{13}
\contentsline {chapter}{O\nobreakspace {}ČESKÉM PŘEKLADU}{27}
\contentsline {chapter}{PŘEDMLUVA W.\,Y. EVANS-WENTZE}{31}
\contentsline {chapter}{ÚVOD}{33}
\contentsline {section}{\numberline {\uppercase {i}}Význam
   Džecün Kambumu}{33}
\contentsline {section}{\numberline {\uppercase {ii}}Historická
   hodnota vyprávění}{34}
\contentsline {part}{\numberline {1}STEZKA TEMNOTY}{67}
\contentsline {chapter}{\cpt@entry {\numberline {\uppercase {i}}PŮVOD
   A\nobreakspace {}NAROZENÍ}{Răčhungův sen vedoucí k\unhbox \voidb@x
   \penalty \@M \ {}napsání tohoto životopisu; \cln Milaräpův původ
   a\unhbox \voidb@x \penalty \@M \ {}narození.}}{67}
\contentsline {chapter}{\cpt@entry {\numberline {\uppercase {ii}}CHUŤ
   STAROSTÍ}{Smrt a\unhbox \voidb@x \penalty \@M \ {}poslední vůle
   Milaräpova otce; protiprávní přisvojení si majetku strýcem
   a\unhbox \voidb@x \penalty \@M \ {}tetou
   z\unhbox \voidb@x \penalty \@M \ {}otcovy strany;
   z\unhbox \voidb@x \penalty \@M \ {}toho vzniklé starosti, jež musel
   Milaräpa snášet se svou matkou
   a\unhbox \voidb@x \penalty \@M \ {}sestrou.}}{78}
\contentsline {chapter}{\cpt@entry {\numberline {\uppercase {iii}}ŠKOLENÍ
```

```
V\nobreakspace {}ČERNÉM UMĚNÍ}{Džecünův guru
a\unhbox \voidb@x \penalty \@M \ {}jeho ovládání černého umění.
Džecün zničí magií pětatřicet svých nepřátel
a\unhbox \voidb@x \penalty \@M \ {}bohatou
žeň ječmene ostatních.}}{85}
```

Zde je pak vidět výsledek.

15.5. Obsah před každou kapitolou

Článek o TEX Live v minulém čísle Zpravodaje [12] postavil redakci před další problém. Článek má totiž vlastní obsah a jeho vypuštěním by došlo ke snížení informační hodnoty. Ladění nových maker je pracné, zvláště když požadovaný účel nabízí balík MINITOC. Ten ovšem zapisuje miniobsah za `\chapter`, ale pohledem do návodu ke psaní článků do Zpravodaje zjistíme, že takové makro se zde nevyskytuje. Souvisí to přirozeně s tím, že používáme třídu ARTICLE. Přesto existuje rychlé a jednoduchě řešení, i když TEXování pak není zcela efektivní.

Nejprve si nadefinujeme čítač pro kapitoly a falešné makro `\chapter`. Čítač inkrementujeme pomocí `\stepcounter`, protože nechceme křížové odkazy na kapitoly. Makro také pouze učiní zápis do pomocného souboru, ale nebude tisknout žádný nadpis. Nakonec zařídíme, aby uživatel ve svém dokumentu nesměl `\chapter` napsat.

```
\newcounter{chapter}
\def\thechapter{\arabic{chapter}}
\def\chapter{\secdef\@chapter\@schapter}
\def\@chapter[#1]#2{\stepcounter{chapter}% no references
  \addcontentsline{toc}{chapter}%
    {\protect\numberline{\thechapter}#1}}
\def\@schapter#1{}
\AtBeginDocument{\let\chapter\undefined} % No real \chapter's
```

Balík MINITOC vyžaduje též přítomnost `\l@chapter` v obsahu. Do pomocného souboru jsme potřebné makro zapsali, ale nechceme nic tisknout. Proto v definici, kterou vidíte níže, použijeme úroveň nadpisu `\@M`, což znamená 10 000. Museli bychom tedy udělat moc velkou chybu, aby se do obsahu tyto záznamy dostaly.

```
\def\l@chapter{\@dottedtocline{\@M}{\z@}{\z@}}
```

Uvedený příklad demonstruje, že při troše znalostí lze i tak zdánlivě neproveditelnou věc lze v LaTeXu naprogramovat za méně než pět minut.

16. Literatura

Prvních pět citací se týká knih a článků, které lze použít k dalšímu studiu a k osvětlení vybraných problémů, o nichž jsme se z různých důvodů nezmínili. Další odkazy se týkají knih, z nichž byly převzaty ukázky.

1. Z. Wagner: *LaTeXová kuchařka/2.* Zpravodaj Československého sdružení uživatelů TeXu, **6** (4), 260–289 (1996).
2. M. Goossens, F, Mittelbach, A. Samarin: *The LaTeX Companion.* Addison Wesley, Reading 1994, ISBN 0-201-54199-8.
3. P. Olšák: *Putování písmene ř z klávesnice na papír.* Zpravodaj Československého sdružení uživatelů TeXu, **7** (3), 129–140 (1977).
4. D. E. Knuth: *The TeXbook.* Addison Wesley, Reading 1984. ISBN 0-201-13448-9.
5. P. Olšák: *TeXbook naruby.* Vyjde v nakladatelství KONVOJ. Elektronická verze je k dispozici na `http://math.feld.cvut.cz/olsak/tbn/`.
6. E. Petiška: *Sagenschatz der böhmischen Burgen.* Martin, 1994.

7. K. Minařík – H. P. Blavacká: *Mahájánské texty.* Canopus, Praha 1995. ISBN 80-85202-26-3.
8. K. Minařík: *Tajemství Tibetu 1.* Sedm tibetských textů. Canopus, Praha 1994, 1996. ISBN 80-85202-23-9.
9. K. Minařík: *Tajemství Tibetu 2.* Tibetská kniha mrtvých (Bardo thödol). Canopus, Praha 1994, 1996. ISBN 80-85202-24-7.
10. W. Y. Evans-Wentz: *Tibetan Yoga And Secret Doctrines.* Oxford University Press, London 1935.
11. W. Y. Evans-Wentz, K. Minařík: *Milaräpa, velký tibetský jógin.* Canopus, Praha 1996. ISBN 80-85202-28-X.
12. S. Rahtz, M. Goossens: *The TEX Live Guide, version 2.* Zpravodaj Československého sdružení uživatelů TEXu, **7** (1–2), 34–88 (1997).

Zdeněk Wagner
wagner@mbox.cesnet.cz

Zápis z valného shromáždění Československého sdružení uživatelů TEXu konaného dne 12. 10. 1997

Valné shromáždění bylo zahájeno ve 14.00 hod na půdě fakulty informatiky MU v Brně. Před tím, v dopoledních hodinách, zde proběhla přednáška Phila Taylora o projektu NTS a o ε-TEXu. Přednášku ve spolupráci s fakultou informatiky uspořádal CSTUG. Vstup byl otevřen pro veřejnost. V kuloárech se rozdávaly pozvánky na EuroTEX98 ve St. Malo, Bretagne.

Valného shromáždění se zúčastnilo 30 řádných členů sdružení, převážně z Brna.

Petr Sojka informoval v rámci zprávy o činnosti o aktivitách jednotlivých členů výboru. Daří se vydávat Zpravodaj díky panu Wagnerovi a časopis dokonce dostal přiděleno ISSN. Rovněž administrativní zázemí CSTUGu se zlepšilo zvláště díky panu ing. Váchovi a paní ing. Váchové. Neplatícím členům sdružení byla zaslána s časopisem 1–2/97 upomínka a připravuje se aktualizace databáze členů. Členství budou zbaveni dlouhodobě neplatící členové a z databáze se odstraní

případně neexistující položky. Výbor se rozhodl zvýšit členské příspěvky individuálním členům vzhledem k postupující inflaci na 120 Kč pro studenty a 240 Kč pro ostatní. Členský příspěvek 1 500 Kč pro kolektivní členy (základní a střední školy platí 500 Kč) zůstává nezměněn. Nová výše členských příspěvků platí od roku 1998. Po třičtvrtěletém úsilí se podařilo převzít agendu po minulém předsedovi CSTUGu. Povedlo se zorganizovat přednášku o typografii doc. Rajlicha v minulém roce a nyní přednášku dr. Philipa Taylora. Nepodařilo se zrealizovat pozvání dr. Franka Mittelbacha. Valné shromáždění zprávu o činnosti schválilo plným počtem hlasů.

Ve zprávě o hospodaření byli přítomní členové seznámeni se stavem účtu CSTUGu a s příjmy a výdaji uskutečněnými v období od 1. 1. do 30. 9. 1997. Na účtu je přes půl miliónu korun, což jsou peníze, které získal CSTUG již dávno v letech minulých při pořádání mezinárodní konference EuroTEX92. Také se ušetřilo z dob, kdy Zpravodaj vycházel nepravidelně. Poslední dva roky byl rozpočet CSTUGu deficitní, protože od roku 1995 bylo schváleno snížení příspěvku kolektivním členům z původních 4 000 Kč měsíčně na 1 000 Kč. Pokud se podaří vydat další dvě čísla Zpravodaje pro letošní rok (číslo 3 už je připraveno k tisku), pak se odhaduje deficit pro letošní rok na 25 tisíc Kč (bez výdajů na TBN). Na druhé straně se letos daří mnohem úspěšněji (za cenu zvýšeného administrativního úsilí) vybírat členské příspěvky. V roce 1997 vstoupilo do řad CSTUGu 99 nových individuálních členů a 1 kolektivní člen. Je registrováno 41 platících (z 59 celkem) kolektivních členů a 297 platících (z 513 celkem) individuálních členů.

Valné shromáždění vzalo tuto zprávu na vědomí s výhradami, že neexistuje zpráva revizora. Proto bude schválení hospodaření CSTUGu odloženo na příští valné shromáždění, až bude existovat zpráva revizora.

Zprávu revizora pana Stanislava Hojka (naposledy volený v roce 1993) za období 1. 1. 1993 až 30. 6. 1996 vzalo shromáždění na vědomí. Dále se přistoupilo k volbě nových revizorů. Pan Wagner navrhl za revizora pana Josefa Ročka a dále padl návrh přijmout do této funkce pana Tomáše Hálu. Oba kandidáti byli přítomni na jednání a svou kandidaturu potvrdili. Podle čl. 8 (Organizační struktura sdružení) stanov jsou revizoři kontrolními orgány a jsou volení valným shromážděním. Jejich počet není ve stanovách zakotven. Valné shromáždění schválilo oba navrhované kandidáty do funkce revizora.

K otázce vydání TEXbooku naruby autora Petra Olšáka předseda Petr Sojka stručně shrnul současný postoj výboru. Na dílo existuje poměrně pozitivní recenze pana Pazdziory a dále je dílo pozitivně přijímano výborem i TEXovskou komunitou. Výbor se rozhodl (na posledním zasedání z 6. 10. 1997) podpořit vydání okamžitým odkoupením 400 ks výtisků z nákladu, pokud bude nakladatel účtovat za jeden kus nejvýše 200 Kč. K tomuto rozhodnutí přispěl výsledek průzkumu zájmu formou závazných objednávek ve Zpravodaji 1–2/97. Po nejlevnější variantě je z tohoto průzkumu patrná okamžitá poptávka po 150 kusech,

dražší varianta by znamenala prodej 130 kusů a nejdražší 110 kusů. Výbor se necítí na to realizovat tento projekt sám jako nakladatel.

S informací o možném nakladateli – edičním středisku MFF UK – vystoupil v krátké zprávě pan Ginzel. Sjednal finanční podporu projektu u zodpovědných činitelů fakulty, ale upozornil, že existuje překážka: knížka by měla jen lepenou vazbu a vhledem k limitu 320 stránek na jednu vazbu by musela vyjít ve dvou dílech. Nejdříve by se o nákladu mohlo uvažovat na jaře příštího roku. Proto připustil, že by bylo lépe, kdyby se nákladu ujal někdo, komu nestojí v cestě tato technická omezení.

Dále vystoupil RNDr. Tomáš Hála, jednatel nakladatelství Konvoj. Oznámil, že poté, co se dozvěděl o rozhodnutí výboru, že odkoupí 400 ks nákladu, jednal z Brna telefonicky na návrh Petra Sojky přímo s autorem Petrem Olšákem (v pátek 10. 10. večer) a dnes před valnou hromadou s ním dohodl všechny technické a další záležitosti. TEXbook naruby vydá a dohodnutý počet kusů CSTUGu odprodá. Bude se jednat o pevnou vazbu V8 (tj. nikoli lepenou V2) a celá kniha bude v jednom svazku. Pokud se vše bude dařit podle plánu, může se knížka dostat k prvním čtenářům už na konci tohoto roku. Přítomní členové reagovali na toto oznámení pochvalným zvukovým efektem.

Petr Olšák uvedl, že sice nebude stahovat zveřejněné varianty textu (`dvi` a `ps`), ale nebude je nadále obnovovat. Zůstanou tedy ve staré verzi, zatímco nová verze, která se objeví v nákladu, bude obsahovat opravy všech dosud známých chyb a dále dva nové příklady, které jsou z pohledu autora důležité. Klikací varianta PDF a soubor tbn.mac bude přesně korespondovat s novou tištěnou verzí, jejíž `dvi` a `ps` už nebude vystavováno.

Dále byl Petrem Sojkou nastíněn rámcový plán sdružení na příští období. Uvedl, že četnost veškerých aktivit naráží na problém v lidských zdrojích. Požádal všechny členy, aby si překontrolovali své údaje o sobě v databázi, zvláště pak e-mailovou adresu. V plánu činnosti uvedl podpoření vydání TBN a rozeslání členům podle závazných objednávek. Nadále se výbor bude pokoušet organizovat různé zajímavé přednášky (pokud mají ostatní členové podněty, jsou vítány) a možná by se mohla podařit i akce pracovně nazvaná víkend s TEXem. Uvažuje se o založení skupiny dobrovolníků kolem TEXu (volunteers). Měl by se dotáhnout problém s CSTEXem, zvláště sloučení používaného `czech/slovak.sty` s makrobalíkem Babel, který je součástí Linuxových distribucí TEXu.

Michal Kubeček referoval o technických problémech s reedicí starých Zpravodajů CSTUGu do elektronické podoby. Protože nejsou k dispozici tehdy použité vzory dělení slov a možná i metriky tehdy použitých úprav fontů, nedaří se text zalomit zcela stejně, jako v originálních časopisech. Také jsou problémy s virtualizací a převodem použitých fontů pro formát Type1, který je ve formátu PDF nutný.

Na shromáždění byl představen zástupce České nadace pro podporu free software a byla přislíbena spolupráce s CSTUGem.

Petr Olšák vystoupil v různém s informací, že v Žitné ulici je již vyprodán Jemný úvod do TEXu (3. vydání překladu Doobova originálu). Ukázalo se, že v Brně u pana Marečka je ještě 213 kusů, takže bude vhodné převést část tohoto množství do Prahy, pokud se v Praze najde redistributor. Dále připomenul starý problém, že totiž neexistuje kvalitní korektor překlepů (spell) pro UNIXovou platformu. Doporučil jednat s autory DOSové verze na jejich podmínkách možnosti implementovat korektor do UNIXu. Informoval přítomné se svým záměrem převést TBN do angličtiny s cílem vytvořit anglickou PDF variantu a požádal shromáždění o případnou pomoc. Konečně požádal, aby mu bylo přiděleno právo zápisu do moderované konference členů sdružení `cstugm`.

Petr Sojka upozornil na to, že databáze členů spravovaná v Budějovicích pozbyde koncem tohoto školního roku své správce (dělali to studenti pod vedením Ládi Lhotky a tito studenti končí studium). Zvažovala se možnost přesunu této databáze do Brna na fakultu informatiky nebo do Prahy na elektrofakultu k Martinu Bílému.

Padla otázka po možnosti nákupu Adobe font library (zhruba 300 tisíc Kč). Přítomní došli k závěru, že to je pro sdružení příliš velká finanční částka a navíc by sdružení muselo tyto fonty počeštit. Počeštění pouze na TEXové úrovni (prostřednictvím virtuálních fontů) je sice možné, ale nemá širší použitelnost mimo TEXovou komunitu. Na druhé straně existuje zpráva o existenci podstatně levnějšího CD-čka DTP Studia, kde by měly být fonty počeštěny a (prý) dobře. Někdo z přítomných členů přislíbil vytvoření recenze tohoto CD-čka ve Zpravodaji.

UU: (Úsměvné Ukončení): Protože Petr Olšák inzeroval v listu `cstex` autogramiádu svého titulu TBN, byl v závěru shromáždění promítnut několikavteřinový šot z filmu Výchova dívek v Čechách (zfilmovaný román Michala Viewegha). Na pozadí zní hlas: Nesnáším autogramiády. Starší paní podává autorovi knížku a praví: Tady mi to prosím podepište. Ne snad, že by to dneska mělo nějakou cenu, ale co když umřete...

Zápis provedl
Petr Olšák

Zápis ověřil
Petr Sojka

Informace výboru Československého sdružení uživatelů TEXu

Výbor sdružení zajistil pro slovenské individuální členy výhodnější způsob placení členských příspěvků. Nabízíme následující možnosti:

1. Platby slovenských členů bude vybírat Janka Chlebíková osobně na Ústavu informatiky č. 13, MFF UK, Mlynská dolina, Bratislava. Je vhodné domluvit si návštěvu předem telefonicky (nové číslo 07/65424000 kl. 396, staré číslo 07/724000) nebo e-mailem (chlebikj@dcs.fmph.uniba.sk).
2. Členské příspěvky lze též zasílat poštovní poukázkou na adresu: Janka Chlebíková, Podzáhradná 19, 821 06 Bratislava.
3. Slovenští členové mohou též platit hotově na schůzích, přednáškách a jiných akcích pořádaných sdružením, nebo při návštěvě České Republiky složit peníze u libovolné pobočky Komerční banky, a. s. na účet č. 34735-021/0100 (vedený u pobočky Brno-město).

Slovenští členové mohou platit příspěvky v českých i slovenských korunách. Výbor připomíná, že výše členských příspěvků za rok 1997 činí 200 Kč (200 Sk) pro individuální členy a 100 Kč (100 Sk) pro studenty. Členské příspěvky na rok 1998 činí 240 Kč a 120 Kč pro studenty.

Členské příspěvky lze zaplatit společně s platbou knihy TEXbook naruby.

Kolektivní členové dostávají 3. vydání CD 4AllTEX. Ostatní zájemci si jej mohou objednat na adrese uvedené v tiráži.

[1] ceny v závorkách jsou pro nečleny sdružení

- ☞ **Sborník** konference **EuroTEX '92**, CSTUG, Praha, tvrdá vazba 50,– Kč (100,– Kč)
- ☞ **Zpravodaje CSTUG** (skladem jsou čísla 2,3,4/1991, 3,4/1992, 1,2/1993, 2,3/1994, 3/96). Cena jednoho Zpravodaje je 10,– Kč pro současné členy CSTUGu, 15,– Kč pro ostatní. Tato nabídka neplatí pro čísla posledního ročníku a po vyčerpání zásob.
- ☞ Zahraniční tituly na objednávku:

Knuth, *Computer and Typesetting Series, Vol. A: The TEX Book*, Addison Wesley, ISBN 0201134489 ..... 1285,– (1472,–)
Knuth, *Computer and Typesetting Series, Vol. B: TEX = The Program*, Addison Wesley, ISBN 0201134373 ..... 1764,– (1960,–)
Knuth, *Computer and Typesetting Series, Vol. C: The METAFONT Book*, Addison Wesley, ISBN 0201134446 ..... 1035,– (1135,–)
Knuth, *Computer and Typesetting Series, Vol. D: METAFONT = The Program*, Addison Wesley, ISBN 0201134381 ..... 1730,– (1931,–)
Knuth, *Computer and Typesetting Series, Vol. E: Computer Modern*, Addison Wesley, ISBN 0201134462 ..... 1730,– (1931,–)
Lamport, *LATEX: a Document Preparation System, 2e*, Addison Wesley, ISBN 0201529831 ..... 1255,– (1398,–)
Goosens, *The LATEX Companion*, Addison Wesley, ISBN 0201541998 1290,– (1435,–)
Goosens, Rahtz, Mittelbach, *The LATEX Graphics Companion*, Addison Wesley, ISBN 0201854694 ..... 1290,– (1435,–)
Kopka, *A Guide to LATEX: Document Preparation for Beginners & Advanced*, Addison Wesley, ISBN 020142777X 1741,– (1934,–)
Walsh, *Making TEX Work*, O'Reilly, ISBN 1565920511 ..... 1388,– (1542,–)
Schwarz, *Introduction to TEX*, Addison Wesley, ISBN 020151141X . 933,– (1037,–)
Eijkhout, *TEX by Topic*, Addison Wesley, ISBN 0201568829 ..... 1764,– (1960,–)
Snow, *TEX for the Beginner*, Addison Wesley, ISBN 0201547996 .. 1146,– (1275,–)
Abrahams, *TEX for the Impatient*, Addison Wesley, ISBN 0201513757 ..... 1126,– (1251,–)
Sznyter, *LATEX for Beginners*, Prentice, ISBN 130310891 ..... 1089,– (1147,–)
Hahn, *LATEX for Everyone*, Prentice, ISBN 136059082 ..... 722,– (803,–)

Při nákupu v prodejně je nutné pro získání slevy člena CSTUGu předložit doklad o zaplacení členského příspěvku. Všechny tituly je možné objednat i na dobírku. Dodací lhůta u zahraniční literatury je 3–4 týdny. Všechny uvedené ceny jsou koncové (s DPH 5 %). Knihy objednávejte na adrese poštou na adrese: Knihkupectví Mareček při FI MU, Botanická 68a, 602 00 Brno ☎ (05)–41 512 435, nebo emailem na adresu `bookorders@cstug.cz`. Přímý odběr je možný v pondělí až pátek, 9:00–12:00, 13:00–17:00.

Bližší informace o sdružení CSTUG si lze vyžádat na adrese CSTUG c/o FI MU, Botanická 68a, 602 00 Brno.
Email: `secretary@cstug.cz`, WWW: `http://www.cstug.cz/`.

Zpravodaj Československého sdružení uživatelů TEXu
ISSN 1211-6661

Vydalo:	Československé sdružení uživatelů TEXu vlastním nákladem jako interní publikaci
Obálka:	Bohumil Bednář
Počet výtisků:	670
Uzávěrka:	30. září 1997
Odpovědný redaktor:	Zdeněk Wagner
Tisk a distribuce:	KONVOJ, spol. s r. o., Berkova 22, 612 00 Brno, tel. 05-740233
Adresa:	CSTUG, c/o FI MU, Botanická 68a, 602 00 Brno
fax:	05–412 125 68
e-mail:	cstug@cstug.cz

Zřízené poštovní aliasy sdružení CSTUG:

`bulletin@cstug.cz`, `zpravodaj@cstug.cz`
korespondence ohledně Zpravodaje sdružení

`board@cstug.cz`
korespondence členům výboru

`cstug@cstug.cz`, `president@cstug.cz`
korespondence předsedovi sdružení

`cstug-members@cstug.cz`
korespondence členům sdružení

`cstug-faq@cstug.cz`
řešené otázky s odpověďmi navrhované k zařazení do dokumentu CSFAQ

`secretary@cstug.cz`, `orders@cstug.cz`
korespondence administrativní síle sdružení, objednávky CD-ROM

`bookorders@cstug.cz`
objednávky tištěné TEXové literatury na dobírku

ftp server sdružení:
`ftp://ftp.cstug.cz/`

www server sdružení:
`http://www.cstug.cz/`

Podávání novinových zásilek povoleno Českou poštou, s. p. OZJM Ředitelství v Brně č. j. P/2–1183/97 ze dne 11. 3. 1997.