Zpravodaj Československého sdružení uživatelů TeXu

Zdeněk Wagner

# LaTeXová kuchařka/1

*Zpravodaj Československého sdružení uživatelů TeXu*, Vol. 6 (1996), No. 2, 96–108

Persistent URL: http://dml.cz/dmlcz/149759

## Literatura

[1] Adriana Benadiková, Štefan Mada a Stanislav Weinlich. *Čárové kódy, automatická identifikace*. Grada 1994, 272 stran. ISBN 80-85623-66-8.

[2] Donald Knuth. *The TEXbook*, díl A z *Computers & Typesetting*. Addison-Wesley, Reading, MA, USA 1986, ix+483 stran. Pevná obálka. ISBN 0-201-13447-0.

[3] Petr Olšák. *Typografický systém TEX*. CSTUG 1995, 270 stran. ISBN 80-901950-0-8.

# LaTeXová kuchařka/1

Zdeněk Wagner

Máte-li přístup na Internet, jistě jste již několikrát zaznamenali diskusi, zda je lepší plain TeX nebo LaTeX. Část této polemiky se dostala i na stránky Zpravodaje č. 4/94. Mezi TeXperty je dost zakořeněn mylný názor, že plain TeX poskytuje principiálně kvalitnější sazbu. Ukážeme velmi snadno, že takový názor správný není.

Na počátku zpracování dokumentu máme ASCII soubor. Ten předložíme programu, v OS/2 a v DOSu je to `tex*.exe`, který si navíc načte tzv. formát, obsahující definice základních maker, a metriky fontů. Při zpracování se makra, použitá v dokumentu, expandují s použitím sekvencí definovaných ve formátu až na TeXovské primitivy. V tomto stadiu není rozhodující, v jakém formátu byla makra definována. Výsledné primitivy jsou vždy stejné, tentýž je i algoritmus řádkového a stránkového zlomu. Typografická kvalita dokumentu je tedy v zásadě nezávislá na volbě formátu.

Při prohlížení vytištěných dokumentů ovšem často dojdete k závěru, že výše uvedené tvrzení neplatí. Příčina zřejmě tkví v nezkušenosti některých uživatelů. Standardní třídy ARTICLE, REPORT a BOOK hýří velikostmi písma pro nadpisy, takže výsledek vypadá dosti humpolácky. V LaTeXu se to snadno podaří každému začátečníkovi, který si přečte, že pro název kapitoly má použít `\chapter`. Plain TeX je zde trochu složitější. Začátečník ještě nezná všechny možnosti, a proto jeho dokumenty

vypadají decentněji. Zůstává zde reálná šance, že bude s kvalitou svých textů spokojen a nebude je v budoucnu proměňovat na vzorníky fontů. Výsledek tedy záleží pouze na tom, zda uživatel dokáže své typografické cítění převést do maker formátu, který si pro svoji práci vybral.

Nyní se již každý může svobodně rozhodnout, který formát bude používat. Zde jsem ve sporu s Karlem Horákem. Preferuji LaTeX, protože vše je v něm již uděláno. Makra pro názvy kapitol zapisují vhodná makra pro tisk obsahu do pomocného souboru, takže pak na začátku či konci dokumentu stačí uvést `\tableofcontents`. Plain TeX nic takového nemá, každý si to ve svém dokumentu musí udělat sám. Je pravda, že život je složitý. Každá kniha má názvy kapitol formátovány jinak, liší se vzhled obsahu, seznamu tabulek, rejstříku, atd. Zdálo by se, že stejně v každé knize musíme vše nadefinovat znovu. Naštěstí jsou všechna standardní makra definována strukturovaně. Pokud uživatel ví, kam zasáhnout, stačí předefinovat poměrně malou část a vlastní mechanismus zůstává beze změny. Vzhled LaTeXového dokumentu se pak změní k nepoznání.

Problém práce s LaTeXem tedy spočívá v tom, že uživatel musí vědět, kam zasáhnout. Tento seriál začal vznikat v mé mysli jako návod pro ty, kdo se standardními LaTeXovými třídami nejsou spokojeni, a nechtějí sami studovat `latex.tex`. Touto cestou jsem se kdysi vydal já, když jsem žádnou literaturu nemohl sehnat. Přestože jsem později svá makra upravil studiem knih, které si mi podařilo koupit, mohou občas být zbytečně těžkopádná.

Počet dílů tohoto seriálu je definován jako $n$, přičemž hodnota $n$ není určena. První pokračování je již zhruba připraveno, počet a zaměření dalších dílů bude záviset na čtenářském ohlasu.

Většina příkladů, které budou v seriálu použity, vychází z maker skutečně použitých při sazbě knih. Makra ovšem byla vytvořena pro specifické prostředí, takže nemusí plně vyhovovat požadavkům všech uživatelů. Kromě toho je nutné důrazně upozornit, že v některých příkladech je kladen důraz na demonstraci makrojazyka, ale nedbá se vůbec na typografickou kulturu. Než použijete nějaký příklad ve své produkci, zamyslete se, zda příslušné makro typograficky zapadá do vzhledu dokumentu.

V následujícím textu se budu úmyslně dopouštět nepřesností. Podrobný výklad by totiž nebyl dostatečně přehledný a zbytečně by uživatele zatěžoval detaily, které pro běžnou práci nepotřebují vědět. Zájemci se pak mohou dovědět více vlastním studiem různých knih.

## 1. Proč psát vlastní makra?

V počátečních odstavcích tohoto článku bylo uvedeno, že pro LaTeX je téměř vše hotovo. Když jsem sám s LaTeXem začínal a s něčím jsem si nevěděl rady, dostával jsem odpovědi typu: „Použijte styl ... , který je k dispozici na ....“ Příslušné styly jsem sehnal, otestoval a zjistil, že mi vyhovují. Pak jsem je použil všechny současně, abych ve svém dokumentu získal vše potřebné. Výsledkem byla zpráva:

```
TeX capacity exceeded
```

Je jistě příjemné, když nějakou práci za vás udělá někdo jiný. Nicméně, není vždy nutné chodit na mravence s kanónem. Často místo složitých stylů postačí jednoduché makro. Programátora pak potěší, že dokázal problém vyřešit sám. Navíc nebude zatěžovat paměť svého počítače horou maker, které nehodlá využít.

Musím přiznat, že ve svých začátcích jsem používal AT 286 a DOS. Tím byla dána paměťová omezení. Nyní provozuji TeX v OS/2 na počítači s 32 MB paměti, takže bych se o paměť starat nemusel. Přesto i nyní dávám přednost vlastním jednoduchým makrům před složitými styly.

Neméně podstatnou úlohu totiž hraje (ne)snadnost získávání stylů. Vše je k dispozici na CTAN. Co ale mám dělat, když potřebuji udělat něco odpoledne doma, výsledek musí být odevzdán příští den a z domova se na síť nedostanu. I to je jeden z důvodů, proč vzniká tento seriál.

## 2. LaTeX 2.09 versus LaTeX 2$_\varepsilon$

Často se uživatelé ptají, proč by měli přejít na LaTeX 2$_\varepsilon$, co jim to přinese nového a jaké budou mít potíže. Pak se potýkají s nedostatkem paměti a uvažují, zda to vůbec mělo smysl.

Řada maker v mé kuchařce pochází z doby, kdy LaTeX 2$_\varepsilon$ ještě neexistoval a fungují s oběma verzemi. Budou zde ovšem presentována i makra, která s verzí 2.09 nechodí.

Pro běžného uživatele nepřináší LaTeX 2$_\varepsilon$ nic převratného. Může dál používat své navyklé způsoby vytváření dokumentů a vše bude fungovat. Výhody ocení zejména programátor maker. Pro uživatele však také poskytuje nové možnosti.

Hlavní novinkou je nová verze New Font Selection Scheme, NFSS2. Máme k dispozici ortogonální systém přepínání jednotlivých atributů

písma, takže `\bfseries` zapne písmo **tučné**, `\itshape` přepne na *kurzívu* a kombinace `\bfseries \itshape` v libovolném pořadí poskytne ***tučnou kurzívu***. Tyto možnosti měla i původní NFSS vytvořená pro LaTeX 2.09. Ta ovšem využívala přepínače `\bf` a `\it`, takže dokumenty pak vypadaly odlišně podle toho, jakým formátem byly zpracovány. Některé dokumenty dokonce zpracovat nešly. NFSS2 tento problém řeší. Původní makra `\bf`, `\it` apod. mají stejné funkce jako v LaTeXu 2.09, nové vlastnosti se dosahují novými makry. Zpracování dokumentu, který byl psán pro starou NFSS, pak vyžaduje speciální styl, kde jsou stará makra předefinována.

Při psaní kurzívou nesmíme zapomenout na kurzívní korekci `\/`. Této starosti se vyhneme použitím nového makra `\textit`. Všechna makra `\text..` totiž automaticky vkládají kurzívní korekce tam, kde jsou potřebné. Uveďme si jako příklad následující větu:

Významné *slovo* sázíme kurzívou.

Tuto větu jsme vysázeli takto:

```
Významné \textit{slovo} sázíme kurzívou.
```

Všimněte si, že kurzívní korekce zde není explicitně zapsána, protože makro `\textit` ji dosadí samo. Výhodou je zdánlivě pouze to, že na kurzívní korekci nemusíme myslet. Podívejme se však znovu na větu a rozhodněme se, že ji trochu doplníme:

Významné *slovo*, které chceme zdůraznit, sázíme kurzívou.

Zde jsem za slovo tištěné kurzívou přidali čárku, takže kurzívní korekce tam být nesmí. Při použití `\it` nebo `\itshape` bychom kurzívní korekci museli odstranit ručně a třeba někdy jindy, až bychom čárku odstranili při další změně formulace, museli znovu dopsat. Makro `\textit` nás této úmorné činnosti ušetří.

Podobnou vlastnost má i makro `\emph`. Můžeme tedy psát:

```
Významné \emph{slovo} sázíme kurzívou.
```

a výsledkem bude: „Významné *slovo* sázíme kurzívou.“ Kurzívní korekce bude opět automaticky doplněna makrem `\emph`.

*Pokud sázíme hlavní text kurzívou, pak se pro vyznačování používá písmo stojaté. Nyní musí přijít kurzívní korekce na začátek. I tento případ bude automaticky ošetřen. Stejně zapsaná věta se nyní vytiskne jako: „Významné* slovo *sázíme kurzívou.“*

Nové možnosti přinášejí i makra pro definici maker. TEX poskytuje primitiv `\def`. Při jeho použití máme plnou kontrolu nad způsobem, jak bude nové makro definováno. Můžeme bez varování předefinovat již existující makro a dokonce i primitiv. V mnoha případech se to hodí. Když se nám podaří předefinovat nějaké důležité makro omylem, nestačíme se divit. Jako domácí úkol si vyzkoušejte následující definici:

```
\def\output{\typeout{Kontrola}}
```

Výsledek bude otřesný. Při použití definičního makra `\newcommand` by se to nestalo. Proto je `\newcommand` vhodný zejména pro začátečníky. LATEX $2_\varepsilon$ navíc dává uživatelům vylepšenou verzi maker `\newcommand` a `\renewcommand`. Vylepšení spočívá v možnosti specifikace nepovinného parametru, který bude uveden v hranatých závorkách. Definujme si například:

```
\newcommand\pokus[1][slovo]{Tiskneme \textit{#1} kurzívou.}
```

Zápisem `\pokus` pak dostaneme: „Tiskneme *slovo* kurzívou.“ Můžeme též napsat `\pokus[cokoliv]` a výsledkem bude: „Tiskneme *cokoliv* kurzívou.“ Je-li parametrů více, můžeme tento trik použít pouze pro první z nich.

Pokud chceme změnit existující definici, musíme použít jiné makro: `\renewcommand`. Co ale máme dělat v případě, chceme-li definovat makro `\pokus` jen za předpokladu, že ještě definováno není? Pokud použijeme `\newcommand` a makro je již definováno, bude se nová definice ignorovat a LATEX oznámí chybu. Cíle bylo dosaženo, ale s chybovým hlášením. Lze to samozřejmě obejít různě, ale LATEX $2_\varepsilon$ má elegantní řešení. Tím je definiční makro `\providecommand`, které má stejnou syntaxi jako `\newcommand`.

Další vymoženosti, které získáte přechodem na LATEX $2_\varepsilon$, budou uváděny postupně v dalších kapitolách a v dalších dílech kuchařky. Kdybychom je chtěli popsat na tomto místě, nezbylo by nám, než vměstnat celý seriál do této jediné kapitoly.

## 3. Nefunguje `\vspace`

Zkušený LATEXista mi snad odpustí, že zde budu vysvětlovat zcela triviální problém. Na otázku, proč `\vspace` vkládá mezeru jinam, než má,

jsem už odpovídal tolikrát, že to stojí za popsání v tomto seriálu.

Nejprve si musíme říci něco o tom, jak TEX zpracovává dokument. Velmi zjednodušeně lze říci, že TEX pracuje ve dvou režimech[1], vertikálním a horizontálním. Na začátku dokumentu je TEX přepnut do vertikálního režimu. Zde se může vyskytovat jakýkoliv vertikální materiál, tj. cokoliv, co se sází ve svislém směru. Nelze sem tedy vkládat žádný text. Jakmile se v dokumentu objeví text nebo jiná horizontální položka, přepne se TEX do horizontálního režimu. Celý text včetně dalších horizontálních položek se načítá až do konce odstavce. Ten se pak zlomí na řádky, které se vloží do hlavního vertikálního seznamu. Potom se TEX přepne do vertikálního režimu.

Je zřejmé, že vertikální mezery nelze vkládat uvnitř odstavce. TEX proto při objevení primitivu `\vskip` ukončí odstavec a potom vloží vertikální mezeru. Někdy ovšem potřebujeme vložit vertikální položku (penaltu nebo svislou mezeru) mezi řádky odstavce. K tomu slouží primitiv `\vadjust`.

LATEXovské makro `\vspace` slouží pro oba účely. Pokud se vyskytuje ve vertikálním režimu, tedy mezi odstavci, expanduje se na `\vskip`. Jako parametr makra lze zadat „gumovou“ mezeru. Příkaz:

```
\vspace{4cm plus 2cm minus 1cm}
```

je ekvivalentní primitivu

```
\vskip 4cm plus 2cm minus 1cm
```

Příkaz vloží mezeru výšky 4 cm. Mezeru lze rozpálit na 6 cm nebo stáhnout na 3 cm.

V horizontálním režimu vkládá `\vspace` mezeru prostřednictvím primitivu `\vadjust`. Pro ilustraci je těsně před tečku, která ukončuje tuto větu, vložen příkaz `\vspace{11dd}`. Všimněte si, že mezera se vloží *za* řádek, na němž je příkaz uveden. Pokud vložíte `\vspace` na nevhodné místo uvnitř odstavce, může TEX udělat mezeru jinde, než bylo zamýšleno. To se stává zejména tehdy, když použijete `\vspace` uprostřed makra. Často se takto vložený `\vspace` dostane na začátek odstavce, takže místo mezery mezi odstavci dostanete mezeru za prvním řádkem odstavce.

[1] Podrobnější výklad viz The TEXbook, kapitola 13.

Podobně se chovají i makra `\smallskip`, `\medskip` a `\bigskip`. Ta byla totiž předefinována pomocí `\vspace` a platí o nich totéž, co bylo uvedeno výše.

Vertikální mezery se automaticky zruší na začátku stránky. Pokud skutečně chcete začít stránku svislou mezerou, musíte za `\vspace` přidat hvězdičku.

## 4. Invalid use of `\spacefactor...`

Toto je běžná chybová zpráva, se kterou se potýká každý začínající LaTeXista a neustále se příslušný dotaz objevuje i v diskusních listech. Pochopení této chyby je nezbytné dříve, než se pustíme do dalších výkladů.

V TeXu má každý znak dva atributy, kód a kategorii. Kategorie znaků jsou shrnuty v tabulce 1. Některé názvy jsou ponechány v angličtině, protože český překlad zřejmě není zaveden.

Určité znaky mají kategorie přiděleny automaticky, některé kategorie jsou předefinovány ve formátu. Primitiv `\catcode` umožňuje změnu kategorie kdekoliv v textu. Tímto mechanismem byl v tabulce vytištěn znak `^`. Změnou kategorie na 12 ztratil svůj speciální význam.

Nebudeme se zde příliš zabývat vysvětlováním kategorií. Připomeneme, že aktivní znak se chová jako makro. Můžeme tedy aktivovat například znak „č“ a definovat jeho význam:

```
\def č{\v{c}}
```

Každý výskyt „č“ v textu pak bude nahrazen příkazem `\v{c}`.

Mnohem důležitější jsou pro nás kategorie 11 a 12. Víme, že makro je uvedeno speciálním „escape“ znakem, za nímž následuje posloupnost písmen nebo jeden nepísmenový znak. První případ jsme již viděli třeba v makru `\vspace`, druhým případem je zúžená mezera `\,`.

Formát i styly obsahují vnitřní makra, která by se v textu dokumentu užívat neměla. Je to zařízeno jednoduchým trikem. Kategorie znaku „@“ je změněna na 11. TeX potom považuje „@“ za písmeno, takže můžete definovat i používat makro `\@startsection`. Ve vlastním dokumentu má „@“ kategorii 12. Pokud tedy napíšete `\@startsection`, bude to TeX interpretovat jako makro `\@` následované znaky `startsection`. `\@` se

Tabulka 1: Kategorie znaků

| Kat. | `Příklad` | Význam |
|---|---|---|
| 0 | `\` | Escape |
| 1 | `{` | Začátek skupiny |
| 2 | `}` | Konec skupiny |
| 3 | `$` | Přepínač matematického režimu |
| 4 | `&` | Tabelátor |
| 5 | `^^M` | Konec řádku |
| 6 | `#` | Parametr |
| 7 | `^` | Exponent |
| 8 | `_` | Index |
| 9 | | Ignorovaný znak |
| 10 | | Mezera |
| 11 | `a` | Písmeno |
| 12 | `1` | Jiný znak |
| 13 | `č` | Aktivní znak |
| 14 | `%` | Komentář |
| 15 | | Neplatný znak |

expanduje na `\spacefactor` a tento primitiv má smysl jen za určitých okolností.

Obvykle se začátečník dozví v odpovědi na svoji otázku, že má předefinovat nějaké makro. Vloží tedy do preambule kód obsahující například `\@startsection`. Zde se ovšem `\spacefactor` vyskytovat nesmí, což způsobí chybu uvedenou v nadpisu. V preambuli se nesmí vyskytovat žádný text, proto LaTeX oznámí:

```
Missing \begin{document}
```

„Chybějící“ `\begin{document}` je přidán a vytištěn text `startsection`. Pokud se dále vyskytují makra, která jsou povolena pouze v preambuli, dostanete další chybové zprávy, přinejmenším na `\begin{document}`.

Makra obsahující „@“ lze tedy definovat a používat pouze tam, kde je „@“ písmenem. Je tomu tak v souborech s příponou `.sty`, které jsou načítány příkazem `\usepackage`. Chcete-li taková makra používat v definicích intenzivně, je vhodné, abyste si napsali vlastní soubor s makry a v preambuli jej načetli makrem `\usepackage`. Pokud potřebujete

v preambuli jen jednu definici, můžete použít `\makeatletter` pro změnu kategorie znaku „@“ na 11. Zpětnou změnu kategorie pak provedete makrem `\makeatother`.

První z uvedených metod je využita i při tvorbě tohoto zpravodaje. Všechny základní definice jsou v CSBUL.STY a hlavní TEXový soubor začíná příkazy:

```
\documentclass[twoside]{article}
\usepackage{csbul}
```

## 5. Vizuální a kontextové značkování

V předchozích kapitolách jsme si naznačili základní problémy, se kterými se potká snad každý začátečník. Nyní již můžeme rozebírat složitější případy. Před vlastním popisem maker se ovšem ještě zastavíme trochu u filozofie LATEXu.

Všichni víme, že zdrojový dokument pro TEX je obyčejný ASCII text. Vzhled dokumentu je popsán řídícími příkazy. Tomuto způsobu popisu stránky říkáme značkování (anglicky markup).

Jestliže chceme napsat slovo **tučně**, použijeme buď zápis {`\bfseries tučně`} nebo `\textbf{tučně}`. V obou případech definujeme, jak se má uvedené slovo vytisknout. Použitý příkaz určuje vizuální vzhled a odpovídá zvolení stylu *tučně* ve WYSIWYG editoru. Tomuto způsobu označování vzhledu části textu říkáme *vizuální značkování*.

Vizuální značky mají svůj pevný význam bez ohledu na to, kde se vyskytují. Objevíme-li při čtení zdrojového textu `\textbf`, víme, že argument makra bude vytištěn tučně. Tento přístup má tedy výhodu v tom, že je zde jasně patrná korespondence mezi příkazem a vzhledem části textu.

Nyní si představme následující situaci. Píšeme článek do sborníku, kde názvy kapitol mají být tučně. Použijeme pro ně `\textbf`. V textu chceme zdůraznit některá slova. Opět použijeme `\textbf`, abychom je vysázeli tučně. Článek odevzdáme a redaktor nám oznámí, že zvýrazněná slova nemají být tučná, ale v kurzívě. Musíme tedy některá `\textbf` přepsat na `\textit`. Ještě horší to může být s nadpisy. Obvykle se nadpis odděluje od předchozího i následujícího textu mezerou. V našem článku to mohlo vypadat například takto:

```
\vspace{22pt plus 11pt minus 6pt}
\noindent\textbf{Nadpis}

\vspace{11pt plus 6pt minus 3pt}
\noindent Začátek odstavce...
```

Nyní si představte, že redator bude vyžadovat jiné mezery nad a pod nadpisem. V takovém případě nezbyde než projít ručně celý dokument a provést mnoho úprav, které jsou nudné a pracné a navíc se při tom dá napáchat spousta chyb. Zde vizuální značkování předvádí názorně své nevýhody.

Problém se snadno vyřeší *kontextovým značkováním*. Základní rozdíl spočívá v tom, že kontextová značka definuje *logický význam* části textu, nikoliv jeho vizuální podobu. Za nejjednodušší kontextovou značku bychom snad mohli považovat `\emph`, protože LaTeXu říká, že příslušná pasáž má být zvýrazněna. Konkrétní způsob zvýraznění záleží na tom, jakým písmem je sázen základní text.

Nesporně kontextovými značkami jsou makra pro sazbu názvů kapitol, podkapitol, apod. Makro `\section` provádí řadu činností. Nejenže vytiskne název kapitoly odpovídajícím stylem, ale navíc může automaticky kapitolu očíslovat, připraví makra pro křížové odkazy, předá informace pro živé záhlaví, zabrání stránkovému zlomu pod názvem kapitoly i za prvním řádkem prvního odstavce, vytvoří záznam pro tisk obsahu. Bylo by velmi pracné a nepohodlné, kdybychom toto vše museli mnohonásobně ručně zapisovat do zdrojového textu.

V konečné fázi nás vždycky zajímá vizuální vzhled. Kontextová značka tedy musí být definována pomocí vizuálních značek. Tento přístup má výhodu zejména v tom, že vzhled všech objektů stejného logického významu snadno udržíme jednotný a v případě potřeby jej můžeme změnit předefinováním jediného makra.

V předchozích příkladech jsme použili `\textbf` jak pro názvy kapitol, tak pro zvýraznění slov v textu. Při čtení zdrojového textu pak snadno zjistíme, která část bude vytištěna tučně. Nemusí však být na první pohled jasné, zda se jedná o název kapitoly či zvýrazněný text.

Při kontextovém značkování použijeme `\section` pro nadpisy a `\duraz` pro zvýrazněný text. Chceme-li zvýrazňovat tučně, použijeme definici:

```
\let\duraz\textbf
```

Po intervenci z redakce, požadující zvýrazňování kurzívou, změníme tuto definici na:

```
\let\duraz\textit
```

nebo:

```
\let\duraz\emph
```

Příklad najdete i v tomto zpravodaji. Ve všech článcích jsou názvy kapitol označeny makrem `\section`. Vizuálně vypadají stejně, pouze v některých článcích jsou kapitoly číslovány. Je toho dosaženo jednoduchou změnou definice, o níž si povíme později.

Podobně je to i s názvem článku. Zdrojový text tohoto příspěvku začíná (zjednodušeně) příkazy:

```
\begin{clanek}
\title{\LaTeX ová kuchařka/1}
\author{Zdeněk Wagner}
\podpis{Zdeněk Wagner\\\texttt{wagner@mbox.cesnet.cz}}
\maketitle
```

Vidíte, že jsou zde použita makra `\title`, `\author` a `\maketitle`. Vizuálně však produkují něco jiného než standardní třída ARTICLE.

Nyní je zřejmá další výhoda kontextového značkování. To nám totiž umožňuje sjednotit způsob označování konkrétních logických objektů. Autor pak píše stejným způsobem článek do časopisu, do sborníku konference i do tohoto zpravodaje.

Představte si, že máte vysázet knihu a rukopis dostanete (dnes již netradičně) napsán na psacím stroji. Písařce bude nějakou dobu trvat, než jej přeťuká do počítače. Grafik bude chvíli přemýšlet nad podobou knihy a vy pak strávíte nějaký čas převodem grafického návrhu do LaTeXových maker. Během té doby již může písařka přepisovat rukopis a používat standardní makra jako `\chapter` a pro kontrolní výtisk použít běžnou třídu BOOK. Mezitím vytvoříte jiný styl, kde bude makru `\chapter` přiřazena jiná vizuální reprezentace.

Kontextové značkování je velmi přínosné i pro redaktory časopisů a sborníků. Časopis má určitou koncepci a měl by být vizuálně jednotný. Pokud autoři důsledně používají kontextové značkování, spočívá sjednocování v předefinování poměrně malého množství maker. Je-li ovšem

článek zaplněn vizuálními značkami, nezbývá, než se v textu důkladně povrtat.

LaTeX je dosti silně založen právě na kontextovém značkování. Proto je paradoxně jednodušší přesazení typograficky ohavného LaTeXového článku s důsledně kontextovým značkováním než sjednocení dobře vysázeného článku v plain TeXu, který se hemží vizuálními makry `\vskip`, `\sevenrm` a jinými.

Vizuální značkování však má své použití. Sázíte-li tiráž nebo titulní stránku knihy, bylo by nesmyslné zdržovat se vývojem kontextových maker. Pokud ale hodláte sázet celou ediční řadu s jednotným vzhledem, vyplatí se vytvoření obecného stylu s definicemi kontextových maker pro titulní stranu, tiráž, vydavatelský záznam, jakož i další části knihy.

## 6. Změna vzhledu LaTeXových dokumentů

Oklikou jsme se dostali znovu tam, kde jsme před jedenácti stranami začínali. V celém seriálu nás bude zajímat vizuální vzhled dokumentu. U článků a knih budeme chtít obsah, ale graficky jinak vyvedený. Chceme tedy použít `\chapter`, `\section`, `\tableofcontents`, ale rádi bychom změnili vizuální reprezentaci. V dalších dílech si tedy řekneme, jakým způsobem toho lze snadno dosáhnout.

Autor má již poměrně rozsáhlý archiv definic použitých při sazbě různých knih. Další makra vznikají při tvorbě tohoto zpravodaje. To vše poslouží jako ilustrace možností LaTeXu. Výklad rozhodně nebude čistě akademický, ale bude založen na příkladech, které byly skutečně použity.

V příštím díle si povíme něco o možnosti použití jiných fontů než Computer Modern, o změně vzhledu obsahu a možná i o dalších problémech.

## 7. Doporučená literatura

První díl byl zároveň posledním, který vysvětloval primitivní záležitosti jako `\vspace` a `\@`. Autor očekává, že se čtenáři sami za domácí úkol naučí základy LaTeXu například z dokumentů, které jsou dodávány jako součást balíku CsTeX.

Pro vážné zájemce o LaTeX bych doporučil knihu: Michel Goossens, Frank Mittelbach, Alexander Samarin – *The LaTeX Companion.* Addison

Wesley, Reading 1994, ISBN 0-201-54199-8. Seriál „LaTeXová kuchařka“ ovšem vzniká nezávisle na této knize. Některé části mohou být duplikovány, avšak v seriálu se bude psát i o tom, co ve zmíněné knize nenajdete. Naproti tomu LaTeX Companion popisuje leccos, o čem se v seriálu psát nebude.

Uvedli jsme, že uživatel plain TeXu má plnou kontrolu nad veškerými mechanismy. To se projevuje také tím, že pomocí primitivu `\def` lze definovat složitá makra, na která `\newcommand` nestačí. Pokud chcete využívat LaTeX na 100 %, musíte si stejně prostudovat základní TeXovou bibli, tedy: Donald E. Knuth – *The TeXbook*. Addison Wesley, Reading 1984. ISBN 0-201-13448-9.

*Zdeněk Wagner*
`wagner@mbox.cesnet.cz`

*Následující dva texty vznikly jako semestrální práce z předmětu „Publikační systém TeX“, který přednáší Petr Olšák na Elektrotechnické fakultě ČVUT. Po drobných, zejména jazykových, úpravách jsou zařazeny do našeho bulletinu, přičemž studenti (autoři jednotlivých příspěvků) s touto formou zveřejnění souhlasí.*

# Instalace TeXu na výkonově „malých“ PC

Jaroslav Řezníček

## 1. Motivace

*„Jó, je to pohoda,“ řekl jsem si, když jsem usedl před klávesnici PC 486, na němž byl již naistalován emTeX. Člověk „to“ jen spustí a už může vesele a relativně rychle pracovat na sazbě větších dokumentů.*

*Představte si však situaci, že vlastníte kromě všelijakého jiného haraburdí i nějaké to „maličké PC“, jehož hardwarové vybavení je opravdu*