Zpravodaj Československého sdružení uživatelů TeXu

Peter Wilson; Jan Šustek
Mělo by to fungovat VII – Makra

Zpravodaj Československého sdružení uživatelů TeXu, Vol. 28 (2018), No. 1-4, 90–100

Persistent URL: http://dml.cz/dmlcz/150109

Mělo by to fungovat VII – Makra

Peter Wilson

Abstrakt

V článku jsou ukázány další možnosti nastavení tvaru a zarovnání odstavce. Dále jsou zde ukázány různé možnosti definování maker v LaTeXu.

Klíčová slova: LaTeX, odstavec, makro, `\parshape`, `\newcommand`, `\def`.

> The raging waves doth belching upwardcast
> The wretched wrackes that round about doe fleete,
> The silken sayles and glistering golden Mast,
> Lies all to torne and trodden under feete.
>
> *The Ship of safegarde*
> Barnabe Googe

Cílem tohoto seriálu je ukázat čtenáři krátké kousky kódu, které mohou vyřešit některé z jeho problémů. Doufám, že situaci ještě více nezkomplikuji v důsledku mých chyb. Opravy, poznámky a návrhy na změny budou vždy vítány.

Hlavním tématem tohoto dílu je definování maker. Otázky tohoto typu, a to zejména v LaTeXu, se ve skupině `comp.text.tex` vyskytují opravdu často.

> The lines are fallen unto me in pleasant places;
> yea I have a goodly heritage.
>
> *Psalm 16, verse 6*

1. Více o odstavcích

Donald Knuth mi poslal následující zdrojový kód se slovy

> „Vytvořil jsem toto makro, které může být užitečné při kontrole parametrů příkazu `\parshape` předtím, než se v odstavci použije skutečný text.“

```
% parshape.tex
%    autor Don Knuth, duben 2007
% \parshapetest{n} nakreslí n řádků horizontálních linek
%    při aktuálně nastavených parametrech odstavce
%    (například \hangindent, \hangafter, \parshape)
```

Z anglického originálu *Glisterings* [9] přeložil Jan Šustek.

 doi: 10.5300/2018-1-4/90

```
\def\parshapetest#1{%
  \leavevmode%% DEK původně použil \indent
  \count255=1 \loop
    \ifnum\count255<#1
    \null\leaders\hrule\hfil\null\break
    \advance\count255 by 1   \repeat
  \null\leaders\hrule\hfil\hskip-\parfillskip
  \null\par}
```

Bohužel už bylo příliš pozdě na to, abych makro využil při přípravě článku [8], který se zabýval sazbou různě tvarovaných odstavců. Byla to škoda i z toho důvodu, že když jsem použil makro `\parshapetest` na některé uvedené příklady, zjistil jsem, že jsem ne zcela pochopil význam některých parametrů odstavce.

Makro `\parshapetest{`⟨*počet*⟩`}` nakreslí ⟨*počet*⟩ horizontálních linek do jednotlivých řádků podle aktuálně nastavených parametrů odstavce. Na první pohled to nezní nijak užitečně. Nicméně makro může ušetřit spoustu času, protože nemusíme vymýšlet vhodný text, na němž by byl dobře vidět výsledný tvar odstavce.

Vyzkoušel jsem například tento příklad z [8].

```
\begingroup
\hangindent=3pc \hangafter=-2
\parshapetest{4}
\endgroup
```

K mému překvapení byl výsledek jiný, než jsem plánoval.

Co jsem si původně neuvědomil, bylo, že i při nastavení `\hangindent` a `\hangafter` se pořád na začátku odstavce použije `\parindent`. Původně plánovaného výsledku lze docílit následujícím nastavením.

```
\begingroup
\parindent=0pt
\hangindent=3pc \hangafter=-2
\parshapetest{4}
\endgroup
```

Výsledek je již správný.

Dále jsem vyzkoušel příklad s makrem `\hangfrom` ze stejného článku, které se používá k sazbě odstavců s více odsazenými řádky. Makro bylo definováno následovně.

```
\newcommand*{\hangfrom}[1]{%
  \setbox\@tempboxa\hbox{{#1}}%
  \hangindent \wd\@tempboxa
  \noindent\box\@tempboxa}
```

Při použití[1]

```
\hangfrom{$\Rightarrow$\space}
\parshapetest{3}
```

dostaneme

$\Rightarrow$

Následuje nastavení zajímavějšího tvaru odstavce a ukázka jeho otestování.

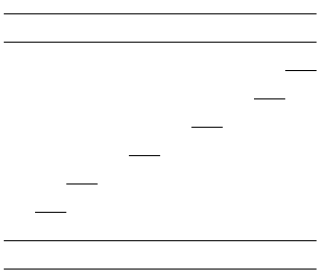
```
\newdimen\delka \delka=\baselineskip
\newcommand*{\zparshape}{%
  \parshape=10 0pt 10\delka % 1
               0pt 10\delka % 2
               9\delka \delka % 3
               8\delka \delka % 4
               6\delka \delka % 5
               4\delka \delka % 6
               2\delka \delka % 7
               \delka \delka % 8
               0pt 10\delka % 9
               0pt 10\delka} % 10
\zparshape
\noindent\parshapetest{10}
```

Výstup je na obrázku 1.

Zkuste si vysázet odstavec s tímto nastavením a textem ze 76 znaků „z“.

```
\zparshape
\noindent
z z z z z z z z z z z z z z z z z z
atd.
```

[1] Všimněte si, že při použití příkazů `\hangindent`, `\hangafter` a `\parshape` není nutné příklady uzavírat mezi `\begingroup` a `\endgroup`. Argumenty těchto příkazů se totiž na konci každého odstavce nulují. (pozn. překl.)



Obrázek 1: Výstup řádků 41–42

Ve skupině `comp.text.tex` položil Stephen Moye dotaz, jak vysázet první řádek odstavce zarovnaný doleva, další řádky zarovnané na střed a poslední řádek zarovnaný doprava. Paul Vojta [7] v odpovědi definoval následující makro, které toto zarovnání nastaví.[2]

```
47 \newcommand*{\leftcenterright}{%
48   \leftskip=0pt plus 1fil
49   \rightskip=0pt plus 1fil
50   \parfillskip=0pt plus -1fil
51   \parindent=0pt
52   \everypar={\hskip0pt plus -1fil\relax}}
```

Jsou dvě možnosti, jak toto makro použít. První z nich je makro použít uvnitř skupiny. Druhou je použít následující makro, které nastaví zarovnání zpět. (Je třeba předem mít nastaven registr `\myparindent` na běžnou hodnotu `\parindent`.)

```
53 \newcommand*{\regularpar}{%
54   \leftskip=0pt
55   \rightskip=\leftskip
56   \parfillskip=0pt plus 1fil
```

[2]Nastavení jednotlivých registrů v makru `\leftcenterright` stojí za vysvětlení.

Na prvním řádku odstavce se vlevo vloží mezery z `\leftskip` a `\parindent` následované obsahem `\everypar`. Celkově tak bude vlevo mezera o velikosti `0pt`. Vpravo bude mezera z `\rightskip`, tj. `0pt plus 1fil`. Nekonečná pružnost mezery vpravo způsobí, že text bude zarovnaný doleva.

Na dalších řádcích odstavce se pouze vloží vlevo mezera z `\leftskip` a vpravo mezera z `\rightskip`. Protože obě mezery jsou stejné a mají nekonečnou pružnost, natáhnou se na stejnou šířku a příslušné řádky budou zarovnané na střed.

Na posledním řádku odstavce se vlevo vloží mezera z `\leftskip`, tj. `0pt plus 1fil`. Vpravo se vloží mezery z `\parfillskip` a `\rightskip`. Celkově tak vpravo bude mezera o velikosti `0pt`. Nekonečná pružnost mezery vlevo způsobí, že text bude zarovnaný doprava.

V původním řešení na řádku 52 chyběl příkaz `\relax`. Cvičením pro pokročilého čtenáře je zjistit, proč je na tomto místě příkaz `\relax` nezbytný. (pozn. překl.)

```
\parindent=\myparindent
\everypar{}}
```

Následuje odstavec vysázený s použitím makra `\leftcenterright`.

```
\leftcenterright
První řádek\break druhý řádek\break
třetí řádek\break poslední řádek\par
\regularpar
```

První řádek

druhý řádek
třetí řádek

poslední řádek

> Who will change old lamps for new?
> ... new lamps for old ones?
> *Arabian Nights: The History of Aladdin*

2. Definiční triumvirát v LaTeXu

LaTeX nabízí pro definování nových maker své makro `\newcommand`, které má trochu jednodušší syntaxi než příkaz TeXu `\def`, na němž je založeno. Syntaxe je

```
\newcommand{⟨název⟩}[⟨počet⟩][⟨arg1⟩]{⟨tělo⟩}
```

kde ⟨*název*⟩ je název definovaného makra včetně zpětného lomítka (například `\makro`) a ⟨*tělo*⟩ je tělo definice makra. To může být jednoduchý text, který se má vysázet, ale může to být také něco velmi složitého. Volitelný parametr ⟨*počet*⟩ určuje, kolik parametrů bude mít nové makro. Pokud se ⟨*počet*⟩ použije, musí mít hodnotu mezi 1 a 9. Pokud je použito ⟨*arg1*⟩, bude první parametr nového makra volitelný a pokud uživatel tento první parametr nepoužije, nastaví se ⟨*arg1*⟩ jako jeho hodnota. Makro definované pomocí `\newcommand` je, řečeno terminologií TeXu, makro typu `\long`, což znamená, že jeho argument může být delší než jeden odstavec nebo, což je ekvivalentní, může obsahovat příkaz `\par`. Varianta s hvězdičkou (`\newcommand*`) nadefinuje makro, které není typu `\long` a jehož argument nesmí obsahovat konec odstavce. Pokud makro ⟨*název*⟩ již dříve bylo definováno, ohlásí LaTeX chybu.

LaTeXové makro

```
\renewcommand{⟨název⟩}[⟨počet⟩][⟨arg1⟩]{⟨tělo⟩}
```

a jeho varianta `\renewcommand*` se chovají obdobně. Jediný rozdíl je, že makro ⟨*název*⟩ musí již dříve být definováno a nyní se předefinuje. Pokud dříve definováno nebylo, ohlásí LaTeX chybu.

Třetí LaTeXové makro pro definování maker je

```
\providecommand{⟨název⟩}[⟨počet⟩][⟨arg1⟩]{⟨tělo⟩}
```

a jeho varianta `\providecommand*`. Pokud makro ⟨*název*⟩ nebylo dříve definováno, chová se `\providecommand` stejně jako `\newcommand`. Pokud makro ⟨*název*⟩ bylo definováno, pak `\providecommand` neudělá nic a neohlásí chybu.

Někdy potřebujete nové makro nadefinovat nezávisle na tom, zda již předtím bylo či nebylo definováno. V tom případě je možné v LaTeXu použít

```
% zajistíme, že makro ⟨název⟩ je definováno
\providecommand{⟨název⟩}{}
% změníme definici
\renewcommand{⟨název⟩}[⟨počet⟩][⟨arg1⟩]{⟨tělo⟩}
```

Pokud se v těle definice vyskytují argumenty, pak se první z nich označuje `#1`, druhý `#2` až devátý `#9`. Argumenty mohou být použity v libovolném pořadí a mohou se libovolně opakovat.

Čas od času se ve skupině `comp.text.tex` objeví otázka, jak definovat makro s více než devíti argumenty. Řešením je rozdělit makro na dvě nebo více maker a každé z nich načte pouze část argumentů. Řekněme, že chceme makro s jedenácti argumenty. Pak použijeme

```
\newcommand{\jedenact}[9]{%
  % načteme 9 argumentů jako #1 až #9
  \zbytek}
\newcommand{\zbytek}[2]{%
  % desátý argument načteme jako #1
  % jedenáctý argument načteme jako #2
  }
```

Uživatel zavolá makro `\jedenact` zdánlivě s jedenácti argumenty. Přitom ve skutečnosti makro `\jedenact` načte pouze prvních devět argumentů a poté zavolá makro `\zbytek`, které zpracuje zbývající dva argumenty.

Pokud je třeba uvnitř makra `\zbytek` zpracovat například čtvrtý argument, lze jej makru `\zbytek` přenést následovně.

```
\newcommand{\jedenact}[9]{%
  % načteme 9 argumentů jako #1 až #9
  \zbytek{#4}}
\newcommand{\zbytek}[3]{%
  % čtvrtý argument načteme jako #1
  % desátý argument načteme jako #2
  % jedenáctý argument načteme jako #3
  }
```

Pro přenesení argumentu makru `\zbytek` lze také využít příkaz `\def` popsaný v následující sekci.

```
\newcommand{\jedenact}[9]{%
  % načteme 9 argumentů jako #1 až #9
  \def\argIV{#4}%
  \zbytek}
\newcommand{\zbytek}[2]{%
  % čtvrtý argument je uložen v makru \argIV
  % desátý argument načteme jako #1
  % jedenáctý argument načteme jako #2
  }
```

Výše uvedeným způsobem lze samozřejmě načíst libovolný počet argumentů. Při větším počtu argumentů se však tento postup stává nepřehledným. Úplně jiný přístup pak nabízí balíček keyval [3] nebo jeho pozdější rozšíření xkeyval [2]. Pomocí těchto balíčků je možné si jednotlivé argumenty makra pojmenovat a uživatel pak může makro zavolat s libovolným počtem těchto argumentů, například jen s některými z nich.

> He who can properly define and divide is to be considered a god.
>
> *Novum Organum*
> Francis Bacon quoting Plato

3. Diktátor v TEXu

TEX disponuje velmi obecným příkazem `\def` pro definování nových maker. Ukázat všechny jeho možnosti v tomto krátkém článku není možné. Knuthův TEXbook [5, kap. 20] nabízí úplný popis, užitečné však mohou být i knihy Eijkhouta [4, kap. 11] nebo Abrahamse a dalších [1, kap. 4 a 9], které jsou pro začínající uživatele lépe čitelné.

Syntaxe příkazu `\def` je úplně jiná, než na co jsou LATEXoví uživatelé zvyklí.

`\def`⟨*název*⟩⟨*parametry*⟩`{`⟨*tělo*⟩`}`

Stejně jako v případě LATEXu je ⟨*název*⟩ název definovaného makra včetně zpětného lomítka (například `\makro`) a ⟨*tělo*⟩ je tělo definice makra. Povšimněte si, že kolem ⟨*název*⟩ se nepoužívají složené závorky.

Část ⟨*parametry*⟩ popisuje výskyt parametrů makra ⟨*název*⟩. Zde se parametry označují `#1`, `#2` atd., musejí být číslovány ve vzestupném pořadí a navíc je rozdíl, zda mezi parametry použijeme mezeru, nebo ne. Pro srovnání uvádíme dva ekvivalentní řádky kódu.

```
\newcommand*{\makro}[2]{...} % LaTeX
\def\makro#1#2{...}          % TeX
... \makro{něco}{bla} ...    % (La)TeX
```

Pokud chceme, aby argument mohl obsahovat konec odstavce, pak musíme makro definovat jako makro typu `\long`. Navíc TEX nijak uživatele neinformuje, že příslušné makro již bylo definováno. Stará definice se jednoduše nahradí novou. Na toto je třeba dát si pozor, protože se může stát, že předefinujete nějaké důležité makro, o kterém jste ani nevěděli, že existuje. Opět uvádíme dva ekvivalentní řádky kódu.

```
\renewcommand{\makro}[2]{...}      % LaTeX
\long\def\makro#1#2{...}           % TeX
... \makro{Odstavec\par}{text} ... % (La)TeX
```

Pokud se v ⟨*parametry*⟩ vyskytují pouze parametry (tj. `#1` atd.), nazývají se tyto parametry neseparované. Odpovídají přesně LATEXovým povinným parametrům. Na druhou stranu, pokud se za `#`n vyskytuje něco jiného než další parametr nebo složená závorka zahajující ⟨*tělo*⟩, pak se parametr `#`n označuje jako separovaný. Příslušné další znaky (přesněji tokeny) se nazývají separátor. Při zavolání makra se příslušný argument načítá tak dlouho, než TEX narazí na tento separátor. Mechanismus volitelných parametrů v LATEXu interně používá právě separované parametry.

Předpokládejme, že chceme nadefinovat makro `\vlak`, které se bude volat

```
\vlak číslo(h:m)
```

kde `číslo`, `h` a `m` jsou argumenty makra `\vlak`. Výše uvedené LATEXové nástroje takové makro nadefinovat neumožňují. Na druhou stranu, příkaz `\def` ano. Pokud nadefinujeme

```
\def\vlak#1(#2:#3){Vlak~#1 jede v~#2:#3.}
```

a zavoláme

```
\textit{\vlak EC130(8:41)}
```

dostaneme výsledek *Vlak EC130 jede v 8:41.*

Může se stát, že budete potřebovat nadefinovat makro, které má dvě varianty, podobně jako makro nadefinované pomocí `\newcommand`. K tomu může pomoci makro `\@ifnextchar` nadefinované v jádru `\LaTeX`u. Syntaxi má následující

```
\@ifnextchar⟨znak⟩{⟨ano⟩}{⟨ne⟩}
```

Makro se podívá na následující znak na vstupu, který je různý od mezery. Pokud je tento znak shodný s ⟨*znak*⟩, pak se provede ⟨*ano*⟩, v opačném případě se provede ⟨*ne*⟩. Jádro LATEXu nabízí také makro

```
\@ifstar{⟨ano⟩}{⟨ne⟩}
```

které testuje, jestli následující znak je hvězdička. Pokud je, pak hvězdičku odstraní a provede ⟨*ano*⟩, v opačném případě provede ⟨*ne*⟩. Toto makro je definováno následovně.

```
\long\def\@firstoftwo#1#2{#1}
\def\@ifstar#1{%
  \@ifnextchar *{\@firstoftwo{#1}}}
```

Nyní můžete snadno definovat makra ve variantě s hvězdičkou i bez hvězdičky.

```
\makeatletter  % mimo soubory .cls a .sty
\def\hvezda{%
  \@ifstar{\@hvezdaS}{\@hvezdaBez}}
  % definice varianty s hvězdičkou
\def\@hvezdaS#1#2{S~hvězdičkou (#1) a (#2).}

  % definice varianty bez hvězdičky
\def\@hvezdaBez#1#2{Bez hvězdičky (#1) a (#2).}
\makeatother   % mimo soubory .cls a .sty
```

Naše makro bude mít dvě varianty a každá bude mít dva argumenty. Vyzkoušejme

```
\hvezda*{první}{druhý}
\hvezda{první}{druhý}
```

Dostaneme výsledek
S hvězdičkou (první) a (druhý).
Bez hvězdičky (první) a (druhý).

Pokud namísto hvězdičky chceme použít jiný znak, například otazník, můžeme makro definovat následovně.

```
\makeatletter  % mimo soubory .cls a .sty
\def\otaznik{%
  \@ifnextchar ?{\@otaznikS}{\@otaznikBez}}
  % definice varianty s otazníkem
\def\@otaznikS#1#2#3{S~otazníkem (#2) a (#3).}
  % definice varianty bez otazníku
\def\@otaznikBez#1#2{Bez otazníku (#1) a (#2).}
\makeatother   % mimo soubory .cls a .sty
```

V předchozím příkladě makro `\@ifstar` hvězdičku odstranilo. Tady musíme otazník odstranit sami. To zařídí naše makro `\@otaznikS` tak, že jej načte jako argument `#1`. Vyzkoušejme

```
\otaznik?{první}{druhý}
\otaznik{první}{druhý}
```

Dostaneme výsledek
S otazníkem (první) a (druhý).
Bez otazníku (první) a (druhý).

Nyní si ukážeme způsob, jak lze v LaTeXu nadefinovat makro, které bude mít dva volitelné parametry a jeden povinný parametr. Heiko Oberdiek k tomuto účelu vytvořil balíček twoopt [6]. My si ukážeme jednodušší způsob, který se při těchto nezvyklých situacích může hodit. Výsledkem bude makro `\dvavolitelne`, jehož volitelné parametry budou mít implicitní hodnotu `jedna` a `dvě`.[3]

```
\def\dvavolitelne{%
  \@ifnextchar [{\@dvavol}{\@dvavol[jedna]}}
\def\@dvavol[#1]{%
  \@ifnextchar [%
      {\@@dvavol{#1}}{\@@dvavol{#1}[dvě]}}
\def\@@dvavol#1[#2]#3{1 (#1) 2 (#2) 3 (#3)}
```

Připomínám, že tato makra je třeba definovat v místě, kde LaTeX považuje znak `@` jako písmeno, například uvnitř balíčku nebo po nastavení `\makeatletter`. Po použití

```
\dvavolitelne{ahoj}
\dvavolitelne[hello]{ahoj}
\dvavolitelne[hello][ahoj]{baf}
```

dostaneme
1 (jedna) 2 (dvě) 3 (ahoj)
1 (hello) 2 (dvě) 3 (ahoj)
1 (hello) 2 (ahoj) 3 (baf)

Seznam literatury

[1] Abrahams, Paul W., Berry, Karl, Hargreaves, Kathryn A. *TeX for the Impatient*. Addison-Wesley, 1990.
[2] Adriaens, Hendri. *The xkeyval package*. 2005. Dostupné na CTAN v adresáři `latex/macros/contrib/xkeyval`.
[3] Carlisle, David. *The keyval package*. 1999. Dostupné na CTAN v adresáři `latex/macros/required/graphics`.
[4] Eijkhout, Victor. *TeX by topic, A TeXnician's Reference*. Addison-Wesley, 1991. Dostupné na `www.wijkhout.net/tbt`.
[5] Knuth, Donald E. *The TeXbook*. Addison-Wesley, 1984.

[3] Doporučuji čtenáři, aby si sám na papíru vyzkoušel postupnou expanzi makra ve všech třech případech. (pozn. překl.)

[6] Oberdiek, Heiko. *The twoopt package: Definitions with two optional arguments.* 1999. Dostupné na CTAN v adresáři `latex/macros/contrib/oberdiek`.
[7] Vojta, Paul. *Re: New York Times headline style.* Příspěvek ve skupině `comp.text.tex`, 10. 7. 2007.
[8] Wilson, Peter. Glisterings. *TUGboat*, 28(2):229–232, 2007.
[9] Wilson, Peter. Glisterings. *TUGboat*, 29(2):324–327, 2008.

Summary

This paper demonstrates more possibilities of setting the paragraph shape and alignment in LaTeX. It also shows several possibilities to define macros in LaTeX.

Key words: LaTeX, paragraph, macro, `\parshape`, `\newcommand`, `\def`.

Peter Wilson, herries.press@earthlink.net
18912 8th Ave. SW
Normandy Park, WA 98166 USA