

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>

Terms of use:

© Československé sdružení uživatelů TeXu, 2018

Institute of Mathematics of the Czech Academy of Sciences provides access to digitized documents strictly for personal use. Each copy of any part of this document must contain these *Terms of use*.



This document has been digitized, optimized for electronic delivery and stamped with digital signature within the project *DML-CZ: The Czech Digital Mathematics Library* <http://dml.cz>

Abstrakt

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

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 \LaTeX u, se ve skupině `comp.text.tex` vyskytují opravdu často.

The lines are fallen unto me in pleasant places;
yea I have a goodly heritage.

Psalms 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 parametru příkazu `\parshape` předtím, než se v odstavci použije skutečný text.“

```
1 % parshape.tex
2 %   autor Don Knuth, duben 2007
3 %   \parshapetest{n} nakreslí n řádků horizontálních linek
4 %   při aktuálně nastavených parametrech odstavce
5 %   (například \hangindent, \hangafter, \parshape)
```

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

```

6 \def\parshapetest#1{%
7 \leavevmode%% DEK původně použil \indent
8 \count255=1 \loop
9 \ifnum\count255<#1
10 \null\leaders\hrule\hfil\null\break
11 \advance\count255 by 1 \repeat
12 \null\leaders\hrule\hfil\hskip-\parfillskip
13 \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].

```

14 \begingroup
15 \hangindent=3pc \hangafter=-2
16 \parshapetest{4}
17 \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.

```

18 \begingroup
19 \parindent=0pt
20 \hangindent=3pc \hangafter=-2
21 \parshapetest{4}
22 \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.

```

23 \newcommand*\hangfrom}[1]{%
24   \setbox\@tempboxa\hbox{#{1}}%
25   \hangindent \wd\@tempboxa
26   \noindent\box\@tempboxa}

```


Při použití¹

```

27 \hangfrom{${\Rightarrow}\space}
28 \parshapetest{3}

```

dostaneme

⇒ 

Následuje nastavení zajímavějšího tvaru odstavce a ukázka jeho otestování.

```

29 \newdimen\delka \delka=\baselineskip
30 \newcommand*\zparshape{%
31   \parshape=10 0pt 10\delka % 1
32               0pt 10\delka % 2
33               9\delka \delka % 3
34               8\delka \delka % 4
35               6\delka \delka % 5
36               4\delka \delka % 6
37               2\delka \delka % 7
38               \delka \delka % 8
39               0pt 10\delka % 9
40               0pt 10\delka} % 10
41 \zparshape
42 \noindent\parshapetest{10}

```

Výstup je na obrázku 1.

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

```

43 \zparshape
44 \noindent
45 z z z z z z z z z z z z z z z z
46 atd.

```

¹Všimněte 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
```

²Nastavení 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 řádce 52 chyběl příkaz `\relax`. Cvičením pro pokročilého čtenáře je zjistit, proč je na tomto místě příkaz `\relax` nezbytný. (pozn. překl.)

```

57 \parindent=\myparindent
58 \everypar{}}

```

Následuje odstavec vysázený s použitím makra `\leftcenterright`.

```

59 \leftcenterright
60 První řádek\break druhý řádek\break
61 třetí řádek\break poslední řádek\par
62 \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 \LaTeX

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

```

63 \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 \TeX u, 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.

\LaTeX ové makro

```

64 \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 L^AT_EXové makro pro definování maker je

```
65 \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 L^AT_EXu použít

```
66 % zajistíme, že makro <název> je definováno
67 \providecommand{<název>}{}
68 % změníme definici
69 \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

```
70 \newcommand{\jedenact}[9]{%
71 % načteme 9 argumentů jako #1 až #9
72 \zbytek}
73 \newcommand{\zbytek}[2]{%
74 % desátý argument načteme jako #1
75 % jedenáctý argument načteme jako #2
76 }
```

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.

```
77 \newcommand{\jedenact}[9]{%
78 % načteme 9 argumentů jako #1 až #9
79 \zbytek{#4}}
80 \newcommand{\zbytek}[3]{%
81 % čtvrtý argument načteme jako #1
82 % desátý argument načteme jako #2
83 % jedenáctý argument načteme jako #3
84 }
```

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

```
85 \newcommand{\jedenact}[9]{%
86 % načteme 9 argumentů jako #1 až #9
87 \def\argIV{#4}%
88 \zbytek}
89 \newcommand{\zbytek}[2]{%
90 % čtvrtý argument je uložen v makru \argIV
91 % desátý argument načteme jako #1
92 % jedenáctý argument načteme jako #2
93 }
```

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 \TeX u

\TeX disponuje velmi obecným příkazem `\def` pro definování nových maker. Ukázat všechny jeho možnosti v tomto krátkém článku není možné. Knuthův \TeX book [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 \LaTeX oví uživatelé zvyklí.

```
94 \def<název><parametry>{<tělo>}
```

Stejně jako v případě \LaTeX u 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.


```

95 \newcommand*{\makro}[2]{...} % LaTeX
96 \def\makro#1#2{...}         % TeX
97 ... \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.

```

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

```

Pokud se v \langle parametry \rangle vyskytují pouze parametry (tj. #1 atd.), nazývají se tyto parametry neseparované. Odpovídají přesně \LaTeX ovým povinným parametrům. Na druhou stranu, pokud se za # n vyskytuje něco jiného než další parametr nebo složená závorka zahajující \langle tělo \rangle , 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 \LaTeX u interně používá právě separované parametry.

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

```
101 \vlak číslo(h:m)
```

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

```
102 \def\vлак#1(#2:#3){Vлак~#1 jede v~#2:#3.}
```

a zavoláme

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

dostaneme výsledek *Vлак 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

```
104 \@ifnextchar<znak>{\ano}{\ne}
```

Makro se podívá na následující znak na vstupu, který je různý od mezery. Pokud je tento znak shodný s \langle znak \rangle , pak se provede \langle ano \rangle , v opačném případě se provede \langle ne \rangle . Jádro \LaTeX u nabízí také makro

```
105 \@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.

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

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

```
109 \makeatletter % mimo soubory .cls a .sty
110 \def\hvezda{%
111   \@ifstar{\@hvezdaS}{\@hvezdaBez}}
112 % definice varianty s hvězdičkou
113 \def\@hvezdaS#1#2{S-hvězdičkou (#1) a (#2).}
114 % definice varianty bez hvězdičky
115 \def\@hvezdaBez#1#2{Bez hvězdičky (#1) a (#2).}
116 \makeatother % mimo soubory .cls a .sty
```

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

```
117 \hvezda*{první}{druhý}
118 \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.

```
119 \makeatletter % mimo soubory .cls a .sty
120 \def\otaznik{%
121   \@ifnextchar ?{\@otaznikS}{\@otaznikBez}}
122 % definice varianty s otazníkem
123 \def\@otaznikS#1#2#3{S-otazníkem (#2) a (#3).}
124 % definice varianty bez otazníku
125 \def\@otaznikBez#1#2{Bez otazníku (#1) a (#2).}
126 \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

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

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

Nyní si ukážeme způsob, jak lze v \LaTeX u 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

```
129 \def\dvavolitelne{%
130   \@ifnextchar [{\@dvavol}\@dvavol[jedna]}}
131 \def\@dvavol[#1]{%
132   \@ifnextchar [%
133     {\@@dvavol{#1}}{\@@dvavol{#1}[dvě]}}
134 \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

```
135 \dvavolitelne{ahoj}
136 \dvavolitelne[hello]{ahoj}
137 \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. *T_EX 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. *T_EX by topic, A T_EXnician's Reference*. Addison-Wesley, 1991. Dostupné na `www.wijkhout.net/tbt`.
- [5] Knuth, Donald E. *The T_EXbook*. Addison-Wesley, 1984.

³Doporučuji 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