

Zpravodaj Československého sdružení uživatelů TeXu

Peter Wilson

Mělo by to fungovat. III – Kolize balíčků

Zpravodaj Československého sdružení uživatelů TeXu, Vol. 22 (2012), No. 4, 194–197

Persistent URL: <http://dml.cz/dmlcz/150118>

Terms of use:

© Československé sdružení uživatelů TeXu, 2012

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:

Článek ukazuje několik triků, jak zabránit situaci, kdy dva \LaTeX ové balíčky definují makra se stejným názvem.

Klíčová slova: \LaTeX , kolize názvů maker

All that glisters is not gold —
Often have you heard that told.

Merchant of Venice, Act II scene 7

WILLIAM SHAKESPEARE

Cílem tohoto seriálu je ukázat čtenáři krátké kousky kódu, které mohou vyřešit některé z jeho problémů.

Opravy, poznámky a návrhy na změny budou vždy vítány.

Nedávno se na diskusní skupině `comp.text.tex` objevila otázka, co lze dělat v případě, kdy dojde ke kolizi dvou balíčků, které definují stejně nazvaná makra.

And we are here as on a darkling plain
Swept with confused alarms of struggle and flight,
Where ignorant armies clash by night.

Dover Beach

ALFRED, LORD TENNYSON

1. Kolize dvou balíčků

Jednoduchým způsobem, jak zrušit definici makra, je definovat jej jako makro `\undefined`.

```
1 \let\makro\undefined
```

Makro `\undefined` samozřejmě nemůže být nikde definováno. Jestliže chceme mít větší jistotu, můžeme použít například

```
2 \let\makro\uNDEFiNed
```

nebo jiný nepravděpodobný název.

Přeložil Jan Šustek.

Předpokládejme, že používáme balíčky, řekněme `balicekA` a `balicekB`, které oba definují makro `\makro`. Pokud není druhé `\makro` definováno pomocí `\def`, které pouze přepíše původní definici, ale pomocí L^AT_EXového `\newcommand`, pak si bude L^AT_EX stěžovat, že `\makro` už bylo definováno. Pokud jsou definice v balíčcích `balicekA` a `balicekB` stejné, můžeme problém vyřešit následovně.

```
3 \usepackage{balicekA}
4 \let\makro\undefined
5 \usepackage{balicekB}
```

Protože život bývá složitý, definice se většinou liší. V tomto případě můžeme použít obě dvě definice, ale u první definice musíme změnit název.

```
6 \usepackage{balicekA}
7 \let\makroA\makro
8 \let\makro\undefined
9 \usepackage{balicekB}
```

Po těchto definicích budeme psát `\makroA`, když budeme používat makro z `balickuA` a budeme psát `\makro`, když budeme používat makro z `balickuB`.

Situace se výrazně zkomplikuje, pokud `balicekA` používá `\makro` uvnitř jiného makra, které používáme. V tomto případě musíme doufat, že alespoň v jednom z balíčků autor provede změny, které kolizi zabrání.

2. Kolize třídy a balíčku

Podobný problém nastává, když dojde ke kolizi názvů maker definovaných ve třídě a v balíčku. Setkal jsem se s tím, když jsem vyvíjel třídu `memoir` [4], která obsahuje kód z mnoha balíčků.¹ V několika případech jsem potřeboval mít jistotu, že se konkrétní balíček nenačte, když se používá daná třída. Přišel jsem na následující makro, po kterém si L^AT_EX bude myslet, že některý balíček už byl načten, i když načten nebyl. Argumentem makra je název balíčku.

```
10 \newcommand*{\@memfakeusepackage}[1]{%
11   \@namelet{ver@#1.sty}\@empty
12 \newcommand*{\@namelet}[1]{%
13   \expandafter\let\csname#1\endcsname}
```

(Tento kód musíme vložit v místě, kde se znak `@` chová jako písmeno.)

Jádro L^AT_EXu obsahuje dva užitečné příkazy, které umožňují používat makra, jejichž názvy se nemusejí skládat pouze z písmen.

```
14 \@namedef{<název>}{<definice>}
15 \@nameuse{<název>}
```

¹Jsem autorem většiny z nich.

První z těchto příkazů umožňuje definovat makro nazvané \langle název \rangle a druhý umožňuje toto makro zavolat. Jako příklad uvedu kousek kódu a jeho výsledek. Všimněte si, že nemůžeme přímo volat makro, jehož název obsahuje jiné znaky než písmena.

```
16 \makeatletter
17 \newcommand*{\ru}{are you}
18 \@namedef{ru4me}#1{#1, are you for me?}
19 ‘\ru4me{Fred}’ he asked.\\
20 ‘\@nameuse{ru4me}{Fred}’ he asked.
21 \makeatother
```

‘are you4meFred’ he asked.

‘Fred, are you for me?’ he asked.

Podobně funguje výše definované makro

```
22 \@namelet{\langlenázev\rangle}
```

Namísto primitivu `\def` ale k definování používá primitiv `\let`. Proto, když zavoláme

```
23 \@memfakeusepackage{pack}
```

dostaneme po expanzi

```
24 \let\ver@pack.sty\@empty
```

To je to magické zaklínadlo, po kterém si \LaTeX bude myslet, že jsme balíček `pack` už dříve načteli.

Třída `memoir` obsahuje kód podobný, ale ne identický jako v balíčcích `array`, `dcolumn`, `delarray` a `tabularx`. Použil jsem makro `\@memfakeusepackage`, abych zajistil, že se tyto balíčky nenačtou znovu.

Třída `memoir` také obsahuje kód odpovídající balíčku `ifpdf` Heika Oberdieka [1], avšak neudělal jsem nic pro to, abych zabránil načtení tohoto balíčku. Důsledkem bylo, že se na `comp.text.tex` objevilo vlákno, ve kterém pisatel používal

```
25 \documentclass{memoir}
26 \usepackage{ps4pdf}
```

přičemž \LaTeX nahlásil chybu, že makro `\ifpdf` již bylo definováno. Ukazuje se, že balíček `ps4pdf` načítá balíček `ifpdf`, který definuje makro `\ifpdf`. Toto makro je však definováno také ve třídě `memoir`.

Heiko Oberdiek [2] nabídl jednoduché řešení, podobné řádkům 3–5, ale také následující složitější řešení.

```
27 \documentclass{memoir}
28 % memoir definuje \ifpdf
```

```

29 \makeatletter
30   % uložíme \ifpdf z memoir
31 \let\saved@ifpdf\ifpdf
32   % zrušíme definici \ifpdf
33 \let\ifpdf@\undefined
34   % použijeme balíček ifpdf (definuje \ifpdf)
35 \usepackage{ifpdf}
36   % je \ifpdf nedefinováno?
37 \@ifundefined{ifpdf}{%
38   % pokud ano, použijme uloženou definici z memoir
39   \let\ifpdf\saved@ifpdf
40   % pokud ne, zkontrolujeme, zda jsou definice stejné
41 }{%
42   \ifx\ifpdf\saved@ifpdf
43   \else
44     % definice jsou různé, vypíše se chybové hlášení
45     \latex@error{Ruzny vyznam \@backslash ifpdf}\@ehc
46   \fi
47 }
48 \makeatother
49 \usepackage{ps4pdf}

```

Toto schéma můžeme použít v dalších podobných situacích. Všimněte si, že pokud se jednotlivé definice liší, obdržíme chybové hlášení. To může být velmi užitečné.

Seznam literatury

- [1] Oberdiek, Heiko. The ifpdf package, July 2001. Dostupné na CTAN na [latex/macros/contrib/oberdiek](#).
- [2] Oberdiek, Heiko. Re: memoir, ps4pdf and \ifpdf. Příspěvek na [comp.text.tex](#), 3. 9. 2004.
- [3] Wilson, Peter. Glisterings. *TUGboat*, 25(2):201–202, 2004.
- [4] Wilson, Peter. The memoir class for configurable typesetting, 2004. Dostupné na CTAN na [latex/macros/contrib/memoir](#).

Summary: It Might Work. III. Conflict Between Packages

This paper shows several tricks how to solve the problem when two L^AT_EX packages define macros with the same name.

Keywords: L^AT_EX, conflict between names of macros