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

Mělo by to fungovat. III – Kolize balíčků

Peter Wilson

Abstrakt:

Článek ukazuje několik triků, jak zabránit situaci, kdy dva LaTeXové 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.

doi: 10.5300/2012-4/194

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 LaTeXového `\newcommand`, pak si bude LaTeX stěžovat, že `\makro` už bylo definováno. Pokud jsou definice v balíčcích balicekA a balicekB stejné, můžeme problém vyřešit následovně.

```
\usepackage{balicekA}
\let\makro\undefined
\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.

```
\usepackage{balicekA}
\let\makroA\makro
\let\makro\undefined
\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 LaTeX bude myslet, že některý balíček už byl načten, i když načten nebyl. Argumentem makra je název balíčku.

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

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

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

```
\@namedef{⟨název⟩}{⟨definice⟩}
\@nameuse{⟨název⟩}
```

[1] Jsem autorem většiny z nich.

První z těchto příkazů umožňuje definovat makro nazvané \⟨*název*⟩ 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.

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

‘are you4meFred’ he asked.
‘Fred, are you for me?’ he asked.

Podobně funguje výše definované makro

```
\@namelet{⟨název⟩}
```

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

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

dostaneme po expanzi

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

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

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

```
\documentclass{memoir}
\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 jednoducké řešení, podobné řádkům 3–5, ale také následující složitější řešení.

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

```
\makeatletter
    % uložíme \ifpdf z memoir
\let\saved@ifpdf\ifpdf
    % zrušíme definici \ifpdf
\let\ifpdf\@undefined
    % použijeme balíček ifpdf (definuje \ifpdf)
\usepackage{ifpdf}
    % je \ifpdf nedefinováno?
\@ifundefined{ifpdf}{%
    % pokud ano, použijme uloženou definici z memoir
  \let\ifpdf\saved@ifpdf
    % pokud ne, zkontrolujeme, zda jsou definice stejné
}{%
  \ifx\ifpdf\saved@ifpdf
  \else
    % definice jsou různé, vypíše se chybové hlášení
    \latex@error{Ruzny vyznam \@backslash ifpdf}\@ehc
  \fi
}
\makeatother
\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 LaTeXpackages define macros with the same name.

Keywords: LaTeX, conflict between names of macros