

Zpravodaj Československého sdružení uživatelů TeXu

Jan Šustek

Načítání souboru s argumenty v TeXu

Zpravodaj Československého sdružení uživatelů TeXu, Vol. 25 (2015), No. 1-2, 86–94

Persistent URL: <http://dml.cz/dmlcz/150230>

Terms of use:

© Československé sdružení uživatelů TeXu, 2015

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: Cílem článku je simulovat v T_EXu načítání souboru s argumenty z terminálu. Bude představeno několik řešení tohoto problému, od jednodušších po složitější. Přitom si čtenář připomene některá pravidla T_EXu, s nimiž se tak často nesetká. Jako ukázkou vytvoříme soubor `brozura.tex`, který po zavolání z terminálu `pdftex brozura kniha` načte dokument `kniha.pdf` a přerovná jeho strany do souboru `brozura.pdf`. Ten pak lze vytisknout jako brožuru.

Klíčová slova: T_EX, argumenty, brožura, `scanargs`.

1 Popis problému

Běžně program T_EX voláme s argumentem, kterým je název vstupního souboru. Když zavoláme

```
1 tex vstup
```

bude T_EX zpracovávat vstupní soubor `vstup.tex`. Cílem článku je nasimulovat načítání vstupního souboru s „jeho“ argumenty, například

```
2 tex vstup arg1 arg2
```

Ukažme si nejprve, co se stane, když T_EX zavoláme jako na řádce 2. Znaky

```
3 vstup arg1 arg2
```

jsou T_EXu poslány jako multý vstupní řádek. Input procesor tyto znaky načte jako¹

```
4 \input vstup arg1 arg2$
```

kde `$` je znak s hodnotou `\endlinechar`.

Vstupní soubor obvykle obsahuje makro `\end{document}` nebo `\bye`, případně jinou ukončovací značku. Když hlavní procesor T_EXu dostane příkaz k ukončení, přestane číst další vstupní text a po nezbytných formalitách ukončí svoji činnost. Ke znakům

```
5 arg1 arg2$
```

se tak T_EX nedostane.

¹Příkaz `\input` se provede pouze interně a token `\input` ve čtecí frontě ve skutečnosti nebude.

2 Návrh řešení

Aby se \TeX k uvedeným znakům dostal, nesmí se hlavní procesor při čtení vstupního souboru dostat k žádnému ukončovacímu příkazu. V takovém případě se vstupní soubor celý přečte a čtecí fronta bude pokračovat znaky na řádku 5. \TeX se ale nějak ukončit musí, proto do vstupního souboru musíme ukončovací příkaz vložit trikově a musíme zajistit, aby se k němu hlavní procesor dostal až po přečtení znaků na řádku 5.

Základní myšlenka je jednoduchá – napsat na konec vstupního souboru řádky

```
6 \catcode\endlinechar=2
7 \afterassignment\bye%
8 \def\ARGS{%
```

Nastavení `\catcode` způsobí, že se znak `\` na řádce 5 promění na token `_2` a bude se chovat jako `\`. Nastavení `\afterassignment` způsobí, že se po přiřazení `\def\ARGS` vloží do čtecí fronty token `\bye`. Pokud na konec vstupního souboru napíšeme řádky 6–8, bude čtecí fronta po zavolání řádku 2 a provedení řádku 8 vypadat následovně:

```
9 \def\ARGS{arg1 arg2}\bye
```

A máme, co jsme chtěli. Dostali jsme se k argumentům a trikově jsme ukončili běh \TeX u.²

Problém je, že toto jednoduché řešení nemůže fungovat. Důvodem je konec vstupního souboru uvnitř definice makra `\ARGS`. \TeX totiž při ukončení načítání souboru testuje mimo jiné, zda právě nenačítá definici makra. Pokud k tomu dojde, \TeX předpokládá, že je to chyba uživatele, který zapomněl napsat uzavírací závorku za definicí. V tom případě \TeX nahlásí chybu a ukončí načítání definice.

Řešením je ukončit načítání souboru před zahájením definice. K tomu se použije příkaz `\noexpand` na úplném konci souboru. Příkaz `\noexpand` načte ze čtecí fronty jeden token. Jelikož okamžitě za příkazem končí soubor, je čtení souboru ukončeno a čtecí fronta pokračuje znaky na řádce 5. Příkaz `\noexpand` tak přečte token `_11`. Je víceméně vedlejší, že se tím tokenu přiřadí příznak *no_expand_flag*. Důležité je, že k tomu dojde před zahájením definice. \TeX pak čte tokeny `\def\ARGS{_1_11` z paměti a během čtení definice nenarazí na konec souboru. Zbývá zajistit, aby se \TeX k příkazu `\noexpand` skutečně dostal dříve než k příkazu `\def`. K tomu poslouží příkaz `\expandafter`. Funkční řešení na posledních řádcích vstupního souboru vypadá následovně.

```
10 \catcode\endlinechar=2
11 \afterassignment\bye%
12 \expandafter\def\expandafter\ARGS\expandafter{\noexpand%
```

²Řádky 7 a 8 je nutné zakončit procenty, aby token procesor nevytvořil token `_2` i na koncích těchto řádků.

Jako příklad vytvoříme soubor `vypis.tex`, který po zavolání

```
13 tex vypis nejake argumenty
```

vypíše na terminál

```
14 Argumenty jsou nejake argumenty.
```

Rozdíl oproti řádkům 10–12 bude v tom, že se po přiřazení neukončí běh `TeXu`, ale že se zavolá makro `\RUN`, které vypíše řádek 14 a až poté ukončí běh `TeXu`. Připomínám, že `\bye` je `\outer` makro, a proto se token `\bye` nesmí vyskytnout uvnitř definice jiného makra.

```
15 \def\RUN{\message{Argumenty jsou \ARGS.}}
16 \csname bye\endcsname}
17 \catcode\endlinechar=2
18 \afterassignment\RUN%
19 \expandafter\def\expandafter\ARGS\expandafter{\noexpand%
```

Všimněme si, že pokud budou argumenty obsahovat nějaká makra, vypíšou se tato po úplné expanzi. Čtenář si lehce domyslí, jak tuto expanzi při výpisu potlačit.

3 Využití registrů `\toks`

Řádky 17–19 nevypadají z uživatelského pohledu příliš přívětivě. Lépe by vypadalo, kdybychom si nadefinovali makro `\SCAN` a uživatel by pak na konci souboru psal pouze

```
20 \SCAN
```

Zde ale narazíme na další problém, a sice na nepárovou závorku na řádku 19, která se uvnitř definice makra `\SCAN` nesmí vyskytnout. Problém lze vyřešit pomocí registru `\toks`, který místo otevírací závorky umožňuje použít token `\bgroup`, jenž nepodléhá kontrole na párování závorek. V následujícím řešení je navíc nastavení `\catcode` umístěno do skupiny, aby nemohlo způsobit další komplikace v případě uživatelské neopatrnosti.³

```
21 \def\SCAN{\begingroup \catcode\endlinechar=2
22 \afterassignment\RUN
23 \toks0=\expandafter\bgroup\noexpand}
24 \def\RUN{\xdef\ARGS{the\toks0}\endgroup
25 \message{Argumenty jsou \ARGS.}
26 \csname bye\endcsname}
27 \SCAN%
```

³Na konci řádku 22 už procento být nemusí, protože ke změně `\catcode` dojde až na řádku 27, kdy už je konec řádku 22 dávno zpracován.

Použití `\xdef` na řádce 24 neprovede úplnou expanzi argumentů, ale úplnou expanzi `\the\toks0`. Jejím výsledkem jsou neexpandované tokeny přesně tak, jak je zadal uživatel na terminálu.

Vadou na kráse ovšem stále zůstává znak `%` na řádce 27, který je v tuto chvíli nutný, aby se potlačil znak konce řádku 27, ze kterého by vznikl token `\S_2`. Tím by se ale ukončilo načítání `\toks0` a k argumentům z terminálu bychom se nedostali.

V následujícím řešení tohoto faktu využijeme a odstraněním procenta ukončíme přiřazení do `\toks0`. Na toto místo totiž můžeme vložit další token pomocí `\afterassignment`. A expanzí tohoto tokenu analogicky načteme argumenty z terminálu. Takto dostáváme nový soubor `vypis.tex`, který už uvedenou vadu na kráse nemá.

```

28 \def\SCAN{\begingroup \catcode\endlinechar=2
29   \afterassignment\SCANc
30   \toks0=\bgroup}
31 \def\SCANc{\afterassignment\RUN
32   \toks2=\expandafter\bgroup\noexpand}
33 \def\RUN{\xdef\ARGS{\the\toks2}\endgroup
34   \message{Argumenty jsou \ARGS.}
35   \csname bye\endcsname}
36 \SCAN

```

Pro pochopení se podrobně podívejme, jak se bude postupně měnit čtecí fronta na řádce 36, pokud zavoláme řádek 13. Text na následujících řádcích, který není v rámečcích, je text již přečtený input procesorem, ale zatím nedotčený token procesorem.

```

37 \SCAN$nejake argumenty$
38 \SCAN$nejake argumenty$
39 \begingroup \catcode \endlinechar =_12 2_12 _10
   \afterassignment \SCANc \toks 0_12 =_12 \bgroup $nejake argumenty$
40 \toks 0_12 =_12 \bgroup \S_2 nejake argumenty$
41 \SCANc nejake argumenty$
42 \afterassignment \RUN \toks 2_12 =_12
   \expandafter \bgroup \noexpand nejake argumenty$
43 \toks 2_12 =_12 \bgroup n_11 e_11 j_11 a_11 k_11 e_11 _10
   a_11 r_11 g_11 u_11 m_11 e_11 n_11 t_11 y_11 \S_2
44 \RUN

```

4 Příprava brožury

V dalším příkladu, inspirovaném makry Petra Olšáka (2016), vytvoříme užitečný soubor `brozura.tex`, který po zavolání

načte dokument kniha.pdf a přeuspořádá jeho strany do souboru brozura.pdf, jehož oboustranným vytištěním a přeložením dostaneme brožuru.

Základ souboru je stejný jako v předcházejících příkladech. Rozdíl je jen v makru \RUN. V něm se nejprve příkazem \pdf_{lastximagepages} zjistí počet stránek souboru kniha.pdf a pak se v cyklu po jednoduchém výpočtu vloží příslušné stránky do souboru brozura.pdf. Je použit příkaz \ship_{out}, kterým se obejde algoritmus stránkového zlomu i output rutina a bez nutnosti nastavování záhlaví a zápatí se příslušný box přímo vysází do souboru brozura.pdf.

Soubor brozura.tex vypadá následovně.

```

46 \def\SCAN{\begingroup \catcode\endlinechar=2
47   \afterassignment\SCANc
48   \toks0=\bgroup}
49 \def\SCANc{\afterassignment\RUN
50   \toks2=\expandafter\bgroup\noexpand}
51 \newcount\al \newcount\ar \newcount\bl \newcount\br
52 \def\RUN{\xdef\nazev{\the\toks2}\endgroup
53   \pdfpagewidth=297mm \pdfpageheight=210mm
54   \pdfhorigin=0pt \pdfvorigin=0pt
55   \pdfximage{\nazev.pdf}
56   \al=\pdflastximagepages
57   \advance\al by3 \divide\al by4 \multiply\al by4
58   \ar=1 \bl=2 \br=\al \advance\br by-1
59   \loop
60     \shipout\hbox{\strana\al \strana\ar}
61     \shipout\hbox{\strana\bl \strana\br}
62     \advance\ar by2 \advance\al by-2
63     \advance\br by-2 \advance\bl by2
64   \ifnum\al>\ar
65   \repeat
66   \csname end\endcsname}
67 \def\strana#1{%
68   \ifnum#1>\pdflastximagepages
69     \hbox to.5\pdfpagewidth{\hss}%
70   \else
71     \pdfximage width.5\pdfpagewidth page#1 {\nazev.pdf}%
72     \pdfrefximage\pdflastximage
73   \fi}
74 \SCAN

```

Pokud se soubor `brozura.tex` umístí do adresářové struktury \TeX u, může uživatel pohodlně psát řádek 45 přímo v adresáři, v němž se nachází soubor `kniha.pdf` a v němž se pak vytvoří soubor `brozura.pdf`.

Soubor `brozura.tex` by se samozřejmě mohl vylepšit. Mohl by například testovat, zda náhodou uživatel nezadal název souboru včetně přípony `.pdf`, zda zadaný soubor existuje, a podobně. To ale není náplní tohoto příkladu.

5 Řešení v balíčku

Cílem této části je uložit dříve definovaná makra do balíčku `scanargs.tex`, aby mohl uživatel zprovoznit načítání argumentů z terminálu pouhým načtením balíčku.

```
75 ... \input scanargs ...
76 \bye
```

Naše řešení spočívá v tom, že se zevnitř balíčku `scanargs.tex` načte zbytek hlavního souboru až po závěrečné `\bye`, pak se uloží argumenty z terminálu a vrátí se k načtenému textu. Problém ale je, že pokud text jednou načteme, není pak možné v něm měnit kategorie znaků, což občas uživatelé potřebují. Dále není možné načíst text, který obsahuje nějaké `\outer` makro, například `\newcount`.

Toto lze obejít dvoupřúchodovým načítáním hlavního souboru. Balíček `scanargs.tex` změní kategorie speciálních znaků na 12 a zbytek hlavního souboru přeskočí. Přeskakované znaky bude načítat jako verbatim a jejich separátorem bude `\bye`, konkrétně tokeny `_12 \b_11 _11 \e_11`. Dále se načtou a uloží argumenty z terminálu. Pak se znovu načte hlavní soubor a opět jako verbatim se přeskočí začátek souboru až po text `\input scanargs`, konkrétně po tokeny `_12 \i_11 \n_11 \p_11 \u_11 \t_11 _12 \s_11 \c_11 \a_11 \n_11 \a_11 \r_11 \g_11 \s_11`. Tím činnost balíčku `scanargs.tex` končí a zbytek hlavního souboru se zpracuje obvyklým způsobem, včetně koncového `\bye`.

V následujícím výpisu souboru `scanargs.tex` se navíc na řádcích 87–88 a 91–105 zjišťuje počet argumentů oddělených mezerou a po jednom je ukládá do paměti. Počet argumentů je pak uložen v registru `\ARGSCOUNT` a *i*-tý argument lze získat zavoláním `\ARG[i]`. Při počítání argumentů je jako pomocný separátor použit token `\^A_8`, jehož použití v argumentech je velmi nepravděpodobné.

```
77 \def\SCAN{\begingroup
78   \def\do##1{\catcode'##1=12}\dospecials\SCANa}
79 {\catcode'\|=0 \catcode'\|=12 \long\gdef\SCANa#1\bye{\|SCANb}}
80 \def\SCANb{\endgroup
81   \begingroup\catcode\endlinechar=2
82   \afterassignment\SCANc
83   \toks0=\bgroup}
```

```

84 \def\SCANc{\afterassignment\RUN
85   \toks2=\expandafter\bgroup\noexpand}
86 \def\RUN{\xdef\ARGS{\the\toks2}\endgroup
87   \expandafter\expandafter\expandafter\SAVE
88   \expandafter\ARGS\space^^A%
89   \begingroup\def\do##1{\catcode'\##1=12}\dospecials
90   \expandafter\SKIP\input\jobname\relax}
91 \newcount\ARGSCOUNT
92 \def\SAVE#1 #2^^A{%
93   \def\next{#1}%
94   \ifx\next\empty
95   \else
96     \advance\ARGSCOUNT1
97     \expandafter\def\csname ARG[\the\ARGSCOUNT]\endcsname{#1}%
98   \fi
99   \def\next{#2}%
100  \ifx\next\empty
101    \let\next\relax
102  \else
103    \def\next{\SAVE#2^^A}%
104  \fi
105  \next}
106 \def\ARG[#1]{\csname ARG[#1]\endcsname}
107 {\catcode'\|=0 \catcode'\ =12 \catcode'\|=12
108 |long|gdef|SKIP#1\input scanargs{|endgroup}}
109 \expandafter\SCAN\noexpand

```

Balíček `scanargs.tex` lze načíst víceméně na libovolném místě před prvním použitím makra `\ARG` nebo `\ARGSCOUNT`. V případě, že by uživatel definoval makro se stejným jménem, jaké má nějaké pomocné makro v balíčku `scanargs.tex`, nevadí to, pokud makro definuje až po načtení balíčku.

Balíček `scanargs.tex` bude fungovat, jestliže bude hlavní soubor končit řádkem 76 s nepotlačeným koncem řádku. Bude fungovat i v případě, že za řádkem 76 budou prázdné řádky s komentářem. Pokud ale za řádkem 76 bude následovat prázdný řádek bez komentáře, vznikne z něj token `\S_2`, a argumenty se z terminálu nenačtou.

6 Ukázka použití balíčku

V následujícím jednoduchém příkladu si ukážeme použití balíčku `scanargs.tex`. Vytvoříme dokument, ve kterém budou vypsány jednotlivé argumenty, a to jak v expandovaném, tak neexpandovaném tvaru.


```

110 \input opmac
111 \input scanargs
112 \rightskip0ptplus1fil
113 \def\bezzacatku#1>{}
114 Počet argumentů: \the\ARGSCOUNT\par
115 \tmpnum=0
116 \loop
117   \ifnum\tmpnum<\ARGSCOUNT \advance\tmpnum1
118   Argument~\the\tmpnum: \ARG[\the\tmpnum]\csname par\endcsname
119   Argument~\the\tmpnum\ bez expanze: {\tt
120     \expandafter\expandafter\expandafter\expandafter
121     \expandafter\expandafter\expandafter\bezzacatku
122     \expandafter\expandafter\expandafter\meaning
123     \ARG[\the\tmpnum]}
124     \csname par\endcsname
125 \repeat
126 \bye

```

7 Problém nula argumentů

Nyní se zaměříme na související problém, který se autorovi tohoto článku nepodařilo vyřešit. Týká se všech dříve uvedených způsobů načítání argumentů.

Předpokládejme, že uživatel na terminálu načte soubor `vypis.tex` a nezadá argumenty, tj. zavolá `TeX` pouze

```

127 tex vypis

```

Rozeberme si podrobně tuto situaci. Čtecí fronta se bude vyvíjet následovně.

```

128 \input vypis$

```

Příkaz `\input` si vyžádá tokeny, ze kterých poté složí název souboru.

```

129 v11 y11 p11 i11 s11
130 v11 y11 p11 i11 s11
131 v11 y11 p11 i11 s11
132 v11 y11 p11 i11 s11
133 v11 y11 p11 i11 s11 $11

```

Znak `\endlinechar` má v tuto chvíli (na začátku běhu `TeX`) kategorii 5, proto se znak `$` promění na token \$₁₀.

```

134 v11 y11 p11 i11 s11 $10

```

Token s ASCII-hodnotou 32 slouží jako oddělovač názvu souboru a je hlavním procesorem pohlčen. Čtecí fronta tak zůstane prázdná a důležitý token \$₂ se

nevytvoří. Pak se načte soubor `vypis.tex`. Makra na konci tohoto souboru očekávají, že ve čtecí frontě ještě nějaké znaky jsou. To však není pravda, a proto \TeX vypíše na terminál hvězdičku a tím uživatele vyzve ke vložení znaků.

Uživatel pak píše znaky úplně stejně, jako by je psal na konci řádku 127. To znamená, že se za tyto znaky automaticky vloží token \S_2 . Tím se ukončí načítání `\toks2`, za nímž se již pak makro `\RUN` vykoná obvyklým způsobem.

Pokud na výzvu \TeX u uživatel žádné znaky nenapíše a pouze stiskne enter, pak bude ve čtecí frontě pouze znak \S vložený input procesorem. Z tohoto znaku se stane token \S_2 a vše bude opět v pořádku. Registr `\toks2` a makro `\ARGS` budou prázdné.

Aby uživatel nebyl zmaten hvězdičkou vypsanou na terminálu, bylo by dobré, kdyby existoval způsob, jak za běhu \TeX u zjistit, zda je čtecí fronta prázdná, nebo zda ještě něco obsahuje. Autorovi článku se však takový způsob najít nepodařilo.

Literatura

OLŠÁK, P. *\TeX pro pragmatiky*. Brno: CSTUG, v tisku. 150 s.
ISBN 978-80-901950-1-1.

Summary: Reading Files with Arguments in \TeX

The paper shows a possibility in \TeX how to input a file with arguments from the terminal. There are simple solutions and also more complicated solutions with several advanced rules of \TeX explained. As an application there is file `brochure.tex`. The user inputs this file by `pdftex brochure book`. Then \TeX reads the document `book.pdf` and reorders its pages into file `brochure.pdf` so that it can be printed as a brochure.

Key words: \TeX , arguments, brochure, scanargs.

*Jan Šustek, jan.sustek@osu.cz
Ostravská univerzita, Přírodovědecká fakulta, Katedra matematiky
30. dubna 22, CZ-701 03 Ostrava, Czech Republic*