

The `withargs` package*

Michiel Helvensteijn
mhelvens+latex@gmail.com

December 19, 2016

Development of this package is organized at github.com/mhelvens/latex-withargs.
I am happy to receive feedback there!

1 Introduction

An example is worth a thousand words:

```
\withargs [!] [Hello] [world] { #2 #3#1 }
```

```
Hello world!
```

A quick explanation: we're passing three pieces of \LaTeX code in the form of optional arguments to the final argument, which forms the output. We're defining a new anonymous macro and invoking it right away. Up to seven optional arguments are supported.

1.1 Why is this useful?

One of the main use-cases for this package is to insert the expanded result of a computation into the middle of a big blob of code that *shouldn't* be expanded. This is possible because argument substitution bypasses the expansion process:

```
\def \expectedResult {\bar}
\def \actualResult   {\bas}
\withargs (oo) [\actualResult] [\expectedResult] {
  \texttt{\detokenize{
    The \foo variable resulted in '#1' instead of '#2'!
  }}}
}
```

```
The \foo variable resulted in '\bas ' instead of '\bar '!
```

'(oo)' indicates that both arguments should be expanded 'once'. Any optional \LaTeX 3 style argument specification between parentheses may be specified to control expansion.

*This document corresponds to `withargs` v0.2.0, dated 2016/12/19.

You can define your own commands accepting ‘templates’ with T_EX-style parameters:

<pre> \newcommand{\NewPost}[4]{ % #1: author % #2: title % #3: content % #4: template \withargs (xnn) [#1] [#2] [#3] [#4] } \def\MyPost{ \NewPost{Author}{Title} {Boy, do I have some important stuff to say!} } \begin{tabular}{p{3.5cm}p{3.5cm}p{3.5cm}} \MyPost{\textbf{\hfill#1}\vskip1mm\hrule\vskip1mm\textit{\#3}} & \MyPost{\fbox{\#1: ``\#2''}\par\#3\vskip1mm\hrule} & \MyPost{\#2: \#3\hfill\textit{\#1}} & \end{tabular} </pre>			
<u>Title</u>	<u>Author</u>	<div style="border: 1px solid black; padding: 2px;">Author: “Title”</div>	Title: Boy, do I have some important stuff to say! (<i>Author</i>)
<i>Boy, do I have some important stuff to say!</i>	Boy, do I have some important stuff to say!		

Generally, if speed is not a concern, `\withargs` can be used to make L^AT_EX code more readable in a variety of situations.

2 Document Level Commands

`\withargs` $\overbrace{(\langle argument\ specification \rangle)}^x$ $\overbrace{[\langle 1 \rangle] [\langle 2 \rangle] [\langle 3 \rangle] [\langle 4 \rangle] [\langle 5 \rangle] [\langle 6 \rangle] [\langle 7 \rangle] \{\langle body \rangle\}}^x$

Leaves $\langle body \rangle$ in the output with #1...#7 replaced by optional arguments $\langle 1 \rangle \dots \langle 7 \rangle$.

An optional L^AT_EX3 style $\langle argument\ specification \rangle$ between parentheses can be provided, in which case the arguments undergo expansion as specified before being placed in the $\langle body \rangle$. Here is an example demonstrating the possible expansion types:

<pre> \def\foo{\bar} \def\s{s} \def\bar{ba\s} \def\bas{FOO-BAR-BAS} \withargs (n o f x c v) [\foo] [\foo] [\foo] [\foo] [\foo] [\foo] { \begin{tabular}{llll} n: & \texttt{\detokenize{\#1}} & & % no expansion o: & \texttt{\detokenize{\#2}} & \backslash & % expand once f: & \texttt{\detokenize{\#3}} & & % expand until unexpandable x: & \texttt{\detokenize{\#4}} & \backslash & % exhaustive expansion c: & \texttt{\detokenize{\#5}} & & % \csname conversion v: & \texttt{\detokenize{\#6}} & & % 'c', then 'o' \end{tabular} } </pre>			
n: \foo	o: \bar		
f: ba\s	x: bas		
c: \bas	v: FOO-BAR-BAS		

Note that all spaces in the *argument specification* are ignored. For details about L^AT_EX3 argument specifications, have a look at the following documentation:

<http://www.ctan.org/pkg/exp13>

`\uniquecsname` Perhaps a bit misleading, `\uniquecsname` is not actually defined as a command, but `\withargs` recognizes it as a special marker. If an optional `\withargs` argument consists entirely of `\uniquecsname`, it is replaced by a command sequence name which is guaranteed to be unique... unless you look at the source code (Section 4) and intentionally replicate the naming scheme.

```
\withargs (ccc) [\uniquecsname][\uniquecsname][\uniquecsname] {
  \def#1{A} \let\A#1
  \def#2{B}
  \def#3{C}
  #3#2#1%
}%
(\A)
CBA(A)
```

3 L^AT_EX3 Functions

The L^AT_EX3 functions underlying the functionality of `\withargs` are also available as is.

`\withargs:nnnnnnnnn` $\overbrace{\{\langle 1 \rangle\} \{\langle 2 \rangle\} \{\langle 3 \rangle\} \{\langle 4 \rangle\} \{\langle 5 \rangle\} \{\langle 6 \rangle\} \{\langle 7 \rangle\} \{\langle 8 \rangle\} \{\langle body \rangle\}}$

These functions do pretty much the same thing as the main `\withargs` macro, except that the argument specification is embedded in the function name as per L^AT_EX3 coding convention, so a parameter slot is freed up for custom use. They are slightly faster than `\withargs`, as there is no need to gather optional arguments or do any error checking. The downside is that the `\uniquecsname` marker doesn't work for these.

To use a specific expansion scheme, you have to define a variant:

```
\ExplSyntaxOn
\cs_generate_variant:Nn \withargs:nn {cn}
\withargs:cn {LaTeX} {#1}
\ExplSyntaxOff
LATEX
```

Read the L^AT_EX3 documentation for details.

4 Implementation

We now show and explain the entire implementation from `withargs.sty`.

4.1 Package Info

```

1 \NeedsTeXFormat{LaTeX2e}
2 \RequirePackage{expl3}
3 \ProvidesExplPackage{withargs}{2016/12/19}{0.2.0}
4 {an inline construct for passing token lists as TeX parameters}

```

4.2 Required Packages

We just need some `expl3`-ish packages.

```

5 \RequirePackage{xparse}
6 \RequirePackage{l3regex}

```

4.3 L^AT_EX3 Functions

$\overbrace{\{\langle 1 \rangle\} \{\langle 2 \rangle\} \{\langle 3 \rangle\} \{\langle 4 \rangle\} \{\langle 5 \rangle\} \{\langle 6 \rangle\} \{\langle 7 \rangle\} \{\langle 8 \rangle\} \{\langle body \rangle\}}^{1-8}$
`\withargs:nn`
`\withargs:nnn`
`\withargs:nnnn`
`\withargs:nnnnn`
`\withargs:nnnnnn`
`\withargs:nnnnnnn`
`\withargs:nnnnnnnn`
`\withargs:nnnnnnnnn`

These are the `expl3` API versions of the `\withargs` command. The implementation is quite straight-forward. This technique has to be used by any library command that accepts T_EX-style parameters.

```

7 \cs_new_protected:Nn \withargs:nn {
8   \cs_set:Npn \__withargs:n ##1 {#2}
9     \__withargs:n {#1} }
10 \cs_new_protected:Nn \withargs:nnn {
11   \cs_set:Npn \__withargs:nn ##1##2 {#3}
12     \__withargs:nn {#1}{#2} }
13 \cs_new_protected:Nn \withargs:nnnn {
14   \cs_set:Npn \__withargs:nnn ##1##2##3 {#4}
15     \__withargs:nnn {#1}{#2}{#3} }
16 \cs_new_protected:Nn \withargs:nnnnn {
17   \cs_set:Npn \__withargs:nnnn ##1##2##3##4 {#5}
18     \__withargs:nnnn {#1}{#2}{#3}{#4} }
19 \cs_new_protected:Nn \withargs:nnnnnn {
20   \cs_set:Npn \__withargs:nnnnn ##1##2##3##4##5 {#6}
21     \__withargs:nnnnn {#1}{#2}{#3}{#4}{#5} }
22 \cs_new_protected:Nn \withargs:nnnnnnn {
23   \cs_set:Npn \__withargs:nnnnnn ##1##2##3##4##5##6 {#7}
24     \__withargs:nnnnnn {#1}{#2}{#3}{#4}{#5}{#6} }
25 \cs_new_protected:Nn \withargs:nnnnnnnn {
26   \cs_set:Npn \__withargs:nnnnnnn ##1##2##3##4##5##6##7 {#8}
27     \__withargs:nnnnnnn {#1}{#2}{#3}{#4}{#5}{#6}{#7} }
28 \cs_new_protected:Nn \withargs:nnnnnnnnn {
29   \cs_set:Npn \__withargs:nnnnnnnn ##1##2##3##4##5##6##7##8 {#9}
30     \__withargs:nnnnnnnn {#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8} }

```

4.4 Document Level Command

`__withargs_var:x` $\{\langle argument\ specifier\rangle\}$

This is a convenience command for generating and using a `\withargs:` variant in one go. I only use it for the document-level command, since those users can't roll their own.

`#1` should be the number of optional `\withargs` arguments and `#2` should be a L^AT_EX3 argument specification not longer than `#1` — a prefix.

```

31 \cs_new:Nn \__withargs_var:nn {
32   \cs_generate_variant:cx
33   { withargs : \prg_replicate:nn{#1}{n} n }
34   { #2 n }
35   \use:c {
36     withargs :
37     #2
38     \prg_replicate:nn{#1-\tl_count:n{#2}}{n}
39     n
40   }
41 }

42 \cs_generate_variant:Nn \__withargs_var:nn {nx}
43 \cs_generate_variant:Nn \cs_generate_variant:Nn {cx}

```

`_withargs_process_uniqecname:n` $\{\langle argument\rangle\}$

An `xparse` processor function to pass a unique control sequence name if the argument given was `'\uniquecsname'`.

```

44 \cs_new_protected:Nn \_withargs_process_uniqecname:n{
45   \tl_if_eq:nnTF {#1} {\uniquecsname} {
46     \int_gincr:N \g__with_unique_int
47     \tl_set:Nx \ProcessedArgument
48       { Unique-CS-Name-( \int_use:N\g__with_unique_int ) }
49   }{
50     \tl_set:Nn \ProcessedArgument {#1}
51   }
52 }
53 \int_new:N \g__with_unique_int

```

`__withargs_remove_spaces:n` $\{\langle argument\rangle\}$

An `xparse` processor function to remove all spaces from the argument.

```

54 \cs_new_protected:Nn \__withargs_remove_spaces:n{
55   \tl_set:Nn \ProcessedArgument {#1}
56   \tl_remove_all:Nn \ProcessedArgument {~}
57 }

```

`\withargs` (*<argument specification>*) [*<1>*] [*<2>*] [*<3>*] [*<4>*] [*<5>*] [*<6>*] [*<7>*] {*<body>*}

This is the document version of the `\withargs` command.

```

58 \NewDocumentCommand {\withargs}
59   { >{\__withargs_remove_spaces:n}      D({}) % argument spec
60     >{\__withargs_process_uniquecsname:n} +o    % up to 7 optional args
61     >{\__withargs_process_uniquecsname:n} +o
62     >{\__withargs_process_uniquecsname:n} +o
63     >{\__withargs_process_uniquecsname:n} +o
64     >{\__withargs_process_uniquecsname:n} +o
65     >{\__withargs_process_uniquecsname:n} +o
66     >{\__withargs_process_uniquecsname:n} +o
67                                     +m } { % the body to execute

```

We first check if the argument specification is valid. It has to be between 0 and 7 characters long and each symbol has to be one of ‘noxfcv’. Otherwise: error! The variants ‘N’ and ‘V’ are not supported (yet) because they collect arguments differently than the others, and frankly, I didn’t want to bother.

```

68 \regex_match:nnF {^[noxfcv]{0,7}$} {#1}
69   { \msg_critical:nnn{withargs}{invalid-parameter-specs}{#1} }

```

The next bit counts the number of optional arguments provided using binary search. If #1 specifies *more* arguments than were provided: error!

```

70 \int_set:Nn \l__with_arg_int {
71   \IfNoValueTF {#5}
72     { \IfNoValueTF {#3} { \IfNoValueTF{#2} 0 1 }
73       { \IfNoValueTF{#4} 2 3 } }
74     { \IfNoValueTF {#7} { \IfNoValueTF{#6} 4 5 }
75       { \IfNoValueTF{#8} 6 7 } }
76   }
77
78 \int_compare:nNnT {\tl_count:n{#1}} > {\l__with_arg_int} {
79   \msg_error:nnxxx{withargs}{invalid-parameter-number}
80     { \tl_count:n{#1} }
81     { \int_use:N \l__with_arg_int }
82     { #1 }
83 }

```

We can then call the right variant of `\withargs`:

```

84 \int_case:nnF {\l__with_arg_int} {
85   {1} { \__withargs_var:nx1{#1} {#2} {#9} }
86   {2} { \__withargs_var:nx2{#1} {#2}{#3} {#9} }
87   {3} { \__withargs_var:nx3{#1} {#2}{#3}{#4} {#9} }
88   {4} { \__withargs_var:nx4{#1} {#2}{#3}{#4}{#5} {#9} }
89   {5} { \__withargs_var:nx5{#1} {#2}{#3}{#4}{#5}{#6} {#9} }
90   {6} { \__withargs_var:nx6{#1} {#2}{#3}{#4}{#5}{#6}{#7} {#9} }
91   {7} { \__withargs_var:nx7{#1} {#2}{#3}{#4}{#5}{#6}{#7}{#8}{#9} }
92 }{}

```

```

93 }
94 \int_new:N \l__with_arg_int

```

The following is the error message displayed if the argument specification is ill-formed:

```

95 \msg_new:nnnn{withargs}{invalid-parameter-specs}{
96   The~argument~specification~'#1'~is~not~valid.
97 }{
98   The~argument~specification~should~consist~of~between~one~
99   and~seven~of~the~letters~'n',~'o',~'f',~'x',~'c',~'v'.
100 }

```

This is the error message displayed if the number of provided optional arguments is inconsistent with the provided argument specification.

```

101 \msg_new:nnnn{withargs}{invalid-parameter-number}{
102   You~specified~#1~arguments~but~provided~#2.
103 }{
104   Your~argument~specification~is~'#3',~which~means~you~should~
105   provide~#1~optional~arguments.~However,~you~provided~#2.~
106   You~should~fix~that.
107 }

```

Change History

0.0.1	0.1.0
General: initial version 1	General: adjusted code to new version
0.0.2	of expl3 1
General: renamed package to <code>withargs</code> 1	0.2.0
<code>\withargs</code> : made the first argument optional and delimited by parentheses 6	General: adjusted code to new version
	of expl3 1

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

Symbols	
<code>__withargs:n</code> 8, 9	<code>__withargs:nnnnnnn</code> 26, 27
<code>__withargs:nn</code> 11, 12	<code>__withargs:nnnnnnnn</code> 29, 30
<code>__withargs:nnn</code> 14, 15	<code>__withargs_process_uniquesname:n</code> 44, 44, 60, 61, 62, 63, 64, 65, 66
<code>__withargs:nnnn</code> 17, 18	<code>__withargs_remove_spaces:n</code> . . . 54, 54, 59
<code>__withargs:nnnnn</code> 20, 21	<code>__withargs_var:nn</code> 31, 42
<code>__withargs:nnnnnn</code> 23, 24	<code>__withargs_var:nx</code> 85, 86, 87, 88, 89, 90, 91

<code>_withargs_var:x</code>	31	<code>\NewDocumentCommand</code>	58
C		P	
<code>\cs_generate_variant:cx</code>	32	<code>\prg_replicate:nn</code>	33 , 38
<code>\cs_generate_variant:Nn</code>	42 , 43	<code>\ProcessedArgument</code>	47 , 50 , 55 , 56
<code>\cs_new:Nn</code>	31	<code>\ProvidesExplPackage</code>	3
<code>\cs_new_protected:Nn</code>		R	
..	7 , 10 , 13 , 16 , 19 , 22 , 25 , 28 , 44 , 54	<code>\regex_match:nnF</code>	68
<code>\cs_set:Npn</code> ...	8 , 11 , 14 , 17 , 20 , 23 , 26 , 29	<code>\RequirePackage</code>	2 , 5 , 6
G		T	
<code>\g__with_unique_int</code>	46 , 48 , 53	<code>\tl_count:n</code>	38 , 78 , 80
I		<code>\tl_if_eq:nnTF</code>	45
<code>\IfNoValueTF</code>	71 , 72 , 73 , 74 , 75	<code>\tl_remove_all:Nn</code>	56
<code>\int_case:nnF</code>	84	<code>\tl_set:Nn</code>	50 , 55
<code>\int_compare:nNnT</code>	78	<code>\tl_set:Nx</code>	47
<code>\int_gincr:N</code>	46	U	
<code>\int_new:N</code>	53 , 94	<code>\uniquecsname</code>	3 , 45
<code>\int_set:Nn</code>	70	<code>\use:c</code>	35
<code>\int_use:N</code>	48 , 81	W	
L		<code>\withargs</code>	2 , 58 , 58
<code>\l__with_arg_int</code>	70 , 78 , 81 , 84 , 94	<code>\withargs:nn</code>	7 , 7
M		<code>\withargs:nnn</code>	7 , 10
<code>\msg_critical:nnn</code>	69	<code>\withargs:nnnn</code>	7 , 13
<code>\msg_error:nnxxx</code>	79	<code>\withargs:nnnnn</code>	7 , 16
<code>\msg_new:nnnn</code>	95 , 101	<code>\withargs:nnnnnn</code>	7 , 19
N		<code>\withargs:nnnnnnn</code>	7 , 22
<code>\NeedsTeXFormat</code>	1	<code>\withargs:nnnnnnnn</code>	7 , 25
		<code>\withargs:nnnnnnnnn</code>	7 , 28