The withargs package*

Michiel Helvensteijn
mhelvens+latex@gmail.com

October 11, 2013

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 LaTeX3 style argument specification between parentheses may be specified to control expansion.

*This document corresponds to withargs v0.0.2, dated 2013/10/11.

You can define your own commands accepting 'templates' with TEX-style parameters:

```
\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{#2\hfill#1}\vskip1mm\hrule\vskip1mm\textit{#3}} &
    \MyPost{\fbox{#1: ``#2''}\par#3\vskip1mm\hrule}                  &
    \MyPost{#2: #3\hfill\textit{(#1)}}
\end{tabular}
```

Title **Author**
Boy, do I have some important stuff to say!

Author: "Title"
Boy, do I have some important stuff to say!

Title: Boy, do I have some important stuff to say! *(Author)*

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

2 Document Level Commands

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

(⟨*argument specification*⟩): x — [⟨*1*⟩] [⟨*2*⟩] [⟨*3*⟩] [⟨*4*⟩] [⟨*5*⟩] [⟨*6*⟩] [⟨*7*⟩]: x

Leaves ⟨*body*⟩ in the output with `#1`...`#7` replaced by optional arguments ⟨*1*⟩...⟨*7*⟩.

An optional LaTeX3 style ⟨*argument specification*⟩ between parentheses can be provided, in which case the arguments undergo expansion as specified before being placed in the ⟨*body*⟩. Here is an example demonstrating the possible expansion types:

```
\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}} \\  % expand once
        f: & \texttt{\detokenize{#3}} &   % expand until unexpandable
        x: & \texttt{\detokenize{#4}} \\  % exhaustive expansion
        c: & \texttt{\detokenize{#5}} &   % \csname conversion
        v: & \texttt{\detokenize{#6}}     % 'c', then 'o'
    \end{tabular}
}
```

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 LaTeX3 argument specifications, have a look at the following documentation:

http://www.ctan.org/pkg/expl3

`\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 LaTeX3 Functions

The LaTeX3 functions underlying the functionality of `\withargs` are also available as is.

`\withargs:nnnnnnnnn` {⟨*1*⟩} {⟨*2*⟩} {⟨*3*⟩} {⟨*4*⟩} {⟨*5*⟩} {⟨*6*⟩} {⟨*7*⟩} {⟨*8*⟩} {⟨*body*⟩}

(with braces labelled *x* over the `nnnnnnnn` part and over arguments ⟨*1*⟩–⟨*8*⟩)

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 LaTeX3 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 LaTeX3 documentation for details.

4 Implementation

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

4.1 Package Info

```
\NeedsTeXFormat{LaTeX2e}
\RequirePackage{expl3}
\ProvidesExplPackage{withargs}{2013/10/11}{0.0.2}
  {an inline construct for passing token lists as TeX parameters}
```

4.2 Required Packages

We just need some expl3-ish packages.

```
\RequirePackage{xparse}
\RequirePackage{l3regex}
```

4.3 LaTeX3 Functions

\withargs:nn
\withargs:nnn
\withargs:nnnn
\withargs:nnnnn
\withargs:nnnnnn
\withargs:nnnnnnn
\withargs:nnnnnnnn
\withargs:nnnnnnnnn

$\overbrace{\{\langle 1\rangle\}\ \{\langle 2\rangle\}\ \{\langle 3\rangle\}\ \{\langle 4\rangle\}\ \{\langle 5\rangle\}\ \{\langle 6\rangle\}\ \{\langle 7\rangle\}\ \{\langle 8\rangle\}}^{1-8}\ \{\langle body\rangle\}$

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 TeX-style parameters.

```
\cs_new_protected:Nn \withargs:nn {
  \cs_set:Npn \__withargs:n ##1 {#2}
              \__withargs:n     {#1} }
\cs_new_protected:Nn \withargs:nnn {
  \cs_set:Npn \__withargs:nn ##1##2 {#3}
              \__withargs:nn    {#1}{#2} }
\cs_new_protected:Nn \withargs:nnnn {
  \cs_set:Npn \__withargs:nnn ##1##2##3 {#4}
              \__withargs:nnn   {#1}{#2}{#3} }
\cs_new_protected:Nn \withargs:nnnnn {
  \cs_set:Npn \__withargs:nnnn ##1##2##3##4 {#5}
              \__withargs:nnnn  {#1}{#2}{#3}{#4} }
\cs_new_protected:Nn \withargs:nnnnnn {
  \cs_set:Npn \__withargs:nnnnn ##1##2##3##4##5 {#6}
              \__withargs:nnnnn {#1}{#2}{#3}{#4}{#5} }
\cs_new_protected:Nn \withargs:nnnnnnn {
  \cs_set:Npn \__withargs:nnnnnn ##1##2##3##4##5##6  {#7}
              \__withargs:nnnnnn {#1}{#2}{#3}{#4}{#5}{#6} }
\cs_new_protected:Nn \withargs:nnnnnnnn {
  \cs_set:Npn \__withargs:nnnnnnn ##1##2##3##4##5##6##7   {#8}
              \__withargs:nnnnnnn {#1}{#2}{#3}{#4}{#5}{#6}{#7} }
\cs_new_protected:Nn \withargs:nnnnnnnnn {
  \cs_set:Npn \__withargs:nnnnnnnn ##1##2##3##4##5##6##7##8    {#9}
              \__withargs:nnnnnnnn {#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8} }
```

4.4 Document Level Command

`\__withargs_var:x` {⟨*argument specifier*⟩}

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 LaTeX3 argument specification not longer than `#1` — a prefix.

```
\cs_new:Nn \__withargs_var:nx {
  \cs_generate_variant:cx
    { withargs : \prg_replicate:nn{#1}{n} n }
    { #2 n }
  \use:c {
    withargs :
    #2
    \prg_replicate:nn{#1-\tl_count:n{#2}}{n}
    n
  }
}
```

```
\cs_generate_variant:Nn \cs_generate_variant:Nn {cx}
```

`\__withargs_process_uniquecsname:n` {⟨*argument*⟩}

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

```
\cs_new_protected:Nn\__withargs_process_uniquecsname:n{
  \tl_if_eq:nnTF {#1} {\uniquecsname} {
    \int_gincr:N \g__with_unique_int
    \tl_set:Nx \ProcessedArgument
        { Unique-CS-Name-( \int_use:N\g__with_unique_int ) }
  }{
    \tl_set:Nn \ProcessedArgument {#1}
  }
}
\int_new:N \g__with_unique_int
```

`\__withargs_remove_spaces:n` {⟨*argument*⟩}

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

```
\cs_new_protected:Nn\__withargs_remove_spaces:n{
  \tl_set:Nn       \ProcessedArgument {#1}
  \tl_remove_all:Nn\ProcessedArgument {~}
}
```

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

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

```
\NewDocumentCommand {\withargs}
    { >{\__withargs_remove_spaces:n}         D(){} % argument spec
      >{\__withargs_process_uniquecsname:n} +o     % up to 7 optional args
      >{\__withargs_process_uniquecsname:n} +o
      >{\__withargs_process_uniquecsname:n} +o
      >{\__withargs_process_uniquecsname:n} +o
      >{\__withargs_process_uniquecsname:n} +o
      >{\__withargs_process_uniquecsname:n} +o
                                           +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.

```
  \regex_match:nnF {^[nofxcv]{0,7}$} {#1}
    { \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!

```
  \int_set:Nn \l__with_arg_int {
    \IfNoValueTF {#5}
     { \IfNoValueTF {#3} { \IfNoValueTF{#2} 0 1 }
                         { \IfNoValueTF{#4} 2 3 } }
     { \IfNoValueTF {#7} { \IfNoValueTF{#6} 4 5 }
                         { \IfNoValueTF{#8} 6 7 } }
  }

  \int_compare:nNnT {\tl_count:n{#1}} > {\l__with_arg_int} {
    \msg_error:nnxxx{withargs}{invalid-parameter-number}
      { \tl_count:n{#1} }
      { \int_use:N \l__with_arg_int }
      { #1 }
  }
```

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

```
  \int_case:nnn {\l__with_arg_int} {
    {1} { \__withargs_var:nx1{#1} {#2}                          {#9} }
    {2} { \__withargs_var:nx2{#1} {#2}{#3}                      {#9} }
    {3} { \__withargs_var:nx3{#1} {#2}{#3}{#4}                  {#9} }
    {4} { \__withargs_var:nx4{#1} {#2}{#3}{#4}{#5}              {#9} }
    {5} { \__withargs_var:nx5{#1} {#2}{#3}{#4}{#5}{#6}          {#9} }
    {6} { \__withargs_var:nx6{#1} {#2}{#3}{#4}{#5}{#6}{#7}      {#9} }
    {7} { \__withargs_var:nx7{#1} {#2}{#3}{#4}{#5}{#6}{#7}{#8}{#9} }
  }{}
```

```
}
\int_new:N \l__with_arg_int
```

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

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

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

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

Change History

0.0.1

General: initial version 1

0.0.2

General: renamed package to withargs 1

\withargs: made the first argument optional and delimited by parentheses 6

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

__withargs:n 8, 9
__withargs:nn 11, 12
__withargs:nnn 14, 15
__withargs:nnnn 17, 18
__withargs:nnnnn 20, 21
__withargs:nnnnnn 23, 24
__withargs:nnnnnnn 26, 27
__withargs:nnnnnnnn 29, 30
__withargs_process_uniquecsname:n 43, 43, 59, 60, 61, 62, 63, 64, 65
__withargs_remove_spaces:n . . 53, 53, 58
__withargs_var:nx 31, 84, 85, 86, 87, 88, 89, 90
__withargs_var:x 31

C

\cs_generate_variant:cx 32

\cs_generate_variant:Nn 42
\cs_new:Nn 31
\cs_new_protected:Nn 7, 10, 13, 16, 19, 22, 25, 28, 43, 53
\cs_set:Npn . . . 8, 11, 14, 17, 20, 23, 26, 29

G

\g__with_unique_int 45, 47, 52

I

\IfNoValueTF 70, 71, 72, 73, 74
\int_case:nnn 83
\int_compare:nNnT 77
\int_gincr:N 45
\int_new:N 52, 93
\int_set:Nn 69
\int_use:N 47, 80

L

\l__with_arg_int 69, 77, 80, 83, 93

M

\msg_critical:nnn 68
\msg_error:nnxxx 78
\msg_new:nnnn 94, 100

N

\NeedsTeXFormat 1
\NewDocumentCommand 57

P

\prg_replicate:nn 33, 38
\ProcessedArgument 46, 49, 54, 55
\ProvidesExplPackage 3

R

\regex_match:nnF 67
\RequirePackage 2, 5, 6

T

\tl_count:n 38, 77, 79
\tl_if_eq:nnTF 44
\tl_remove_all:Nn 55
\tl_set:Nn 49, 54
\tl_set:Nx 46

U

\uniquecsname *3*, 44
\use:c 35

W

\withargs *2*, 57, 57
\withargs:nn 7, 7
\withargs:nnn 7, 10
\withargs:nnnn 7, 13
\withargs:nnnnn 7, 16
\withargs:nnnnnn 7, 19
\withargs:nnnnnnn 7, 22
\withargs:nnnnnnnn 7, 25
\withargs:nnnnnnnnn 7, 28