styledcmd

Paolo De Donato

03 August 2022

styledcmd is a LaTeX package that allows you to create and manage different versions of your macro in order to be able to choose the better style for every occasion and avoid rewriting code each time.

1 How can you include it in your project?

You need only to have the file `styledcmd.sty` in your current working directory. Otherwise you can manually install it inside your preferred LaTeX compiler (for example `TeXLive` or `MiKTeX`) in order to make it available for all your projects. Instructions for manually install a package can be found on Internet.

Then once you've added it you can include in your project with this command:

```
\usepackage{styledcmd}
```

2 How do you use it?

You can create a formatted macro via the following command

`\newstyledcmd` `\renewstyledcmd` `\providestyledcmd`

```
\newstyledcmd {⟨macro name⟩} {⟨format name⟩} [⟨number of arguments⟩] {⟨code⟩}
```

You can repeat that command for the same macro in order to create different styles, for example these commands

```
\newstyledcmd{\saluto}{informal}[1]{Hi #1}
\newstyledcmd{\saluto}{formal}[1]{Good morning #1}
```

define the two formats `informal` and `formal` for macro `\saluto`. You can directly use the command `\saluto` and use the default format (the first declared one) or using a specific style by passing it as an optional argument. So commands

```
\saluto{uncle}
\saluto[informal]{uncle}
\saluto[formal]{uncle}
```

will be expanded respectively as Hi uncle, Hi uncle, Good morning uncle. With the same syntax you can use `\renewstyledcmd` and `\providestyledcmd` with the same meaning of `\renewcommand` and `\providecommand` respectively.

3 How do you change the default style?

In order to change the default style (the one used when you don't choose explicitily a format) you need to execute the following command

\setGlobalStyle `\setGlobalStyle {⟨command name⟩} {⟨new default format name⟩}`

For example in order to change the default style of command `\saluto` from `informal` to `formal` you need to execute command `\setGlobalStyle{\saluto}{formal}`. With this command the output of preceding commands will instead be Good morning uncle, Hi uncle, Good morning uncle.

4 Customize parameters with xparse

styledcmd loads automatically the xparse package for internal reasons. You can also define new styled commands with the same syntax used by `\NewDocumentCommand` with the following command

\NewDocStyledCMD
\RenewDocStyledCMD
\ProvideDocStyledCMD
`\NewDocStyledCMD {⟨command name⟩} {⟨format name⟩} {⟨arguments format⟩} {⟨code⟩}`

For example we can create the following two styles

```
\NewDocStyledCMD{\prova}{stylea}{r<>}{Stile 1 #1}
\NewDocStyledCMD{\prova}{styleb}{r<>}{Stile 2 #1}
```

in order to execute

```
\prova<Hello>
\prova[stylea]<Hello>
\prova[styleb]<Hello>
```

which are expanded respectively as Stile 1 Hello; Stile 1 Hello; Stile 2 Hello. Notice that the first optional argument passed to a command defined via `\NewDocStyleCMD` will always be interpreted as a style argument, so you should use another syntax for optional arguments or use a mandatory argument for the first place.

For example this declaration `\NewDocStyledCMD{\bad}{style}{o m}{Bad declaration}` should be avoided since for example `\bad[arg1]{arg2}` will interpret `arg1` as a style name and not as the first optional argument for `\bad`.

5 Implementation

```
⟨*package⟩
⟨@@=stycmd⟩
```

\c__stycmd_cmdproxy_str Proxy used to generate styled commands

```
\str_const:Nx \c__stycmd_cmdproxy_str { \object_address:nn
    { stycmd }{ proxy } }

\proxy_create:nnN { stycmd }{ proxy } \c_object_public_str
\proxy_push_member:Vnn \c__stycmd_cmdproxy_str { default }{ tl }

```

(*End definition for* `\c__stycmd_cmdproxy_str`.)

`\__stycmd_cmd:n` Name of a command bounded to some style.
`\__stycmd_cmd_style:nn`
`\__stycmd_cmd_default:n`

```
\cs_new:Nn \__stycmd_cmd:n
  {
    \object_address:nn{ stycmd }{ entity - #1 }
  }

\cs_new:Nn \__stycmd_cmd_style:nn
  {
    \object_member_adr:nnn{ \__stycmd_cmd:n{ #1 } }{ cmd - #2 }
      { stycmd_void }
  }

\cs_new:Nn \stycmd_void_use:N { #1 }
\cs_new_eq:NN \stycmd_void_use:c \use:c

\cs_new:Nn \__stycmd_cmd_default:n
  {
    \object_member_adr:nn{ \__stycmd_cmd:n{ #1 } } { default }
  }

```

(*End definition for* `\__stycmd_cmd:n`, `\__stycmd_cmd_style:nn`, *and* `\__stycmd_cmd_default:n`.)

`\stycmd_xparsecmd:n` Defines the main macro with `\ProvideDocumentCommand`.

```

\cs_new_protected:Nn \__stycmd_xparsecmd_aux:Nn
  {
    \ProvideDocumentCommand { #1 } { o }
      {
        \IfNoValueTF {##1}
          {
            \object_member_use:nn
              {
                \__stycmd_cmd:n{ #2 }
              }
              {
                default
              }
          }
          {
            \object_member_use:nnn
              {
                \__stycmd_cmd:n{ #2 }
              }
              {
                cmd - ##1
              }
              { stycmd_void }
          }
      }
  }

```

```
\cs_generate_variant:Nn \__stycmd_xparsecmd_aux:Nn { cn }

\cs_new_protected:Nn \stycmd_xparsecmd:n
  {
    \__stycmd_xparsecmd_aux:cn { #1 }{ #1 }
  }

```

(*End definition for* `\stycmd_xparsecmd:n`. *This function is documented on page* ??.)

`\stycmd_pcmcmd:n` Defines the main macro with `\providecommand`.

```

\cs_new_protected:Nn \__stycmd_pcmcmd_aux:Nn
  {
    \providecommand { #1 } [1] []
      {
        \tl_if_empty:nTF {##1}
          {
            \object_member_use:nn
              {
                \__stycmd_cmd:n{ #2 }
              }
              {
                default
              }
          }
          {
            \object_member_use:nnn
              {
                \__stycmd_cmd:n{ #2 }
              }
              {
                cmd - ##1
              }
              { stycmd_void }
          }
      }
  }

\cs_generate_variant:Nn \__stycmd_pcmcmd_aux:Nn { cn }

\cs_new_protected:Nn \stycmd_pcmcmd:n
  {
    \__stycmd_pcmcmd_aux:cn { #1 }{ #1 }
  }

```

(*End definition for* `\stycmd_pcmcmd:n`. *This function is documented on page* ??.)

`\stycmd_expcmd:n` Defines the main macro with `\providecommand` but the style argument is mandatory in order to make the command expandable. To use default style pass an empty argument as style.

```

\cs_new_protected:Nn \__stycmd_expcmd_aux:Nn
```

```
  {
    \providecommand { #1 } [1]
      {
        \tl_if_empty:nTF {##1}
          {
            \object_member_use:nn
              {
                \__stycmd_cmd:n{ #2 }
              }
              {
                default
              }
          }
          {
            \object_member_use:nnn
              {
                \__stycmd_cmd:n{ #2 }
              }
              {
                cmd - ##1
              }
              { stycmd_void }
          }
      }
  }

\cs_generate_variant:Nn \__stycmd_expcmd_aux:Nn { cn }

\cs_new_protected:Nn \stycmd_expcmd:n
  {
    \__stycmd_expcmd_aux:cn { #1 }{ #1 }
  }

```

(*End definition for* `\stycmd_expcmd:n`. *This function is documented on page* **??**.)

`\setGlobalStyle` Change the default style for specified command

```

\NewDocumentCommand{\setGlobalStyle}{m m}
  {
    \__stycmd_chdef:Nn #1 { #2 }
  }

\cs_new_protected:Nn \__stycmd_chdef_named:nn
  {
    \__stycmd_pars:cc
      {
        \object_member_adr:nn
          {
            \__stycmd_cmd:n{ #1 }
          }
          { default }
      }
      {
```

```
        \object_member_adr:nnn
          {
            \__stycmd_cmd:n{ #1 }
          }
          {
            cmd - #2
          }
          { stycmd_void }
      }
  }
\cs_generate_variant:Nn \__stycmd_chdef_named:nn { fn }
\cs_new_protected:Nn \__stycmd_chdef:Nn
  {
    \__stycmd_chdef_named:fn{ \cs_to_str:N #1 }{ #2 }
  }

```

(*End definition for* `\setGlobalStyle`. *This function is documented on page* 2.)

`\stycmd_generate:NNN`
`\stycmd_generate:NN`
`\stycmd_generate_renew:NN`

Declare the styled version `#1` of the macro generator command `#2`. the `_renew` variant requires a preceding declaration

```

\cs_new:Nn \__stycmd_pars:NN
  {
    \tl_gset:Nn #1 { #2 }
  }

\cs_generate_variant:Nn \__stycmd_pars:NN { cc }

\cs_new_protected:Nn \__stycmd_generate_aux:NNnn
  {
    \object_if_exist:nF
      {
        \__stycmd_cmd:n{ #3 }
      }
      {
        \object_create:VnnNN \c__stycmd_cmdproxy_str
          { stycmd }{ entity - #3 }
          \c_object_global_str
          \c_object_public_str

        \__stycmd_pars:cc
          {
            \object_member_adr:nn
              {
                \__stycmd_cmd:n{ #3 }
              }
              { default }
          }
          {
            \object_member_adr:nnn
              {
                \__stycmd_cmd:n{ #3 }
              }
```

```
              {
                cmd - #4
              }
              { stycmd_void }
          }

        #2 { #3 }
      }
      \exp_args:Nc #1
        {
          \object_member_adr:nnn
            {
              \__stycmd_cmd:n{ #3 }
            }
            {
              cmd - #4
            }
            { stycmd_void }
        }

  }

\cs_generate_variant:Nn \__stycmd_generate_aux:NNnn { NNfn }

\cs_new_protected:Nn \__stycmd_generate_aux_cmd:NNNn
  {
    \__stycmd_generate_aux:NNfn #1 #2 { \cs_to_str:N #3 }{ #4 }
  }

\cs_new_protected:Nn \__stycmd_generate_renew_aux:Nnn
  {
      \exp_args:Nc #1
        {
          \object_member_adr:nnn
            {
              \__stycmd_cmd:n{ #2 }
            }
            {
              cmd - #3
            }
            { stycmd_void }
        }

  }

\cs_new_protected:Nn \stycmd_generate:NNN
  {
    \cs_new_protected:Npn #1 ##1 ##2
      {
        \__stycmd_generate_aux_cmd:NNNn #2 #3 ##1 { ##2 }
      }
  }
\cs_new_protected:Nn \stycmd_generate:NN
  {
```

```
    \stycmd_generate:NNN #1 #2 \stycmd_xparsecmd:n
  }


\cs_new_protected:Nn \stycmd_generate_renew:NN
  {
    \cs_new_protected:Npn #1 ##1 ##2
      {
        \__stycmd_generate_renew_aux:Nnn #2 { ##1 }{ ##2 }
      }
  }

```

(*End definition for* `\stycmd_generate:NNN`, `\stycmd_generate:NN`, *and* `\stycmd_generate_renew:NN`. *These functions are documented on page* **??**.)

`\newstyledcmd` `\renewstyledcmd` `\providestyledcmd`

Declare a new macro with the specified style name.

```
\stycmd_generate:NN \newstyledcmd \newcommand
\stycmd_generate_renew:NN \renewstyledcmd \renewcommand
\stycmd_generate:NN \providestyledcmd \providecommand
```

(*End definition for* `\newstyledcmd`, `\renewstyledcmd`, *and* `\providestyledcmd`. *These functions are documented on page* 1.)

`\NewDocStyledCMD` `\RenewDocStyledCMD` `\ProvideDocStyledCMD`

Declare a new styled macro with the `\NewDocumentCommand` syntax.

```
\stycmd_generate:NN \NewDocStyledCMD \NewDocumentCommand
\stycmd_generate_renew:NN \RenewDocStyledCMD \RenewDocumentCommand
\stycmd_generate:NN \ProvideDocStyledCMD \ProvideDocumentCommand
```

(*End definition for* `\NewDocStyledCMD`, `\RenewDocStyledCMD`, *and* `\ProvideDocStyledCMD`. *These functions are documented on page* 2.)

⟨/package⟩