styledcmd

Paolo De Donato

2.0.1 2024/03/07

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⟩} {⟨style name⟩} [⟨number of arguments⟩] {⟨code⟩}
```

it has the same syntax of `\newcommand` except for the ⟨*style name*⟩ argument that specify the style. This macro alone creates commands `\⟨macro name⟩` and `\⟨macro name⟩[`⟨*style name*⟩`]` that both expand as ⟨*code*⟩.

The most important feature is that you can call `\newstyledcmd` multiple times with the same ⟨*macro name*⟩ but different ⟨*style name*⟩, in this way each of `\⟨macro name⟩[`⟨*style name*⟩`]` expands to ⟨*code*⟩ associated to specified ⟨*style name*⟩. Notice that if you don't specify a style with just calling `\metamacro name` then it expands as the first created style, that style is the *default* one for such command.

As an 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`. Once you've created these two styles for `\saluto` you can use it with or without the style name argument, for example these 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 style) you need to execute the following command

`\SetGlobalStyle`

`\SetGlobalStyle {\⟨command name⟩} {⟨new default style 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 `\NewDocStyledCMD` 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 Expandable commands

Coomands created by `\newstyledcmd` doesn't work very well in expansion only context due to the presence of optional style argument. In order to be able to create expandable commands you should instead use

`\newstyledcmdExp`
`\renewstyledcmdExp`
`\providestyledcmdExp`

`\newstyledcmdExp {\⟨macro name⟩} {⟨style name⟩} [⟨number of arguments⟩] {⟨code⟩}`

Despite commands created with `\newstyledcmd` the style name of commands created by `\newstyledcmdExp` are always mandatory and must be passed inside curly braces. In order to use the default style just pass an empty string as style name.

For example this code

```
\newstyledcmdExp{\expCMD}{sty1}{Style 1}
\newstyledcmdExp{\expCMD}{sty2}{Style 2}

\expCMD{}
\expCMD{sty1}
\expCMD{sty2}
```

expand as Style 1 Style 1 Style 2

6 Groups

The group mechanism is very different from styledcmd `1.2` and preceding versions. From `2.0` styled commands can be added to a group in order to change toghether their style. Groups are automatically created when you're tying to add commands to it:

`\AddCMDToGroup`

`\AddCMDToGroup {⟨group name⟩} {⟨comma list of commands⟩}`

You can change the default style for each member of a group with

`\SetGroupStyle`

`\SetGroupStyle {⟨group name⟩} {⟨style name⟩}`

Suppose you've created the following styled commands:

```
\newstyledcmd{\LenUnit}{SI}{metre}
\newstyledcmd{\MasUnit}{SI}{kilo}

\newstyledcmd{\LenUnit}{old}{yard}
\newstyledcmd{\MasUnit}{old}{pound}
```

Commands `\LenUnit`, `\MasUnit` will expand as metre, kilo respectively since `SI` is the default style. We've used `\newstyledcmd` but you could use also `\NewDocStyledCMD`, `\newstyledcmdExp` or any other command generated as in section 7. To add these two commands to group `Units` run

```
\AddCMDToGroup{Units}{\LenUnit, \MasUnit}
```

If you want to set `old` as the default style for these commands just run

```
\SetGroupStyle{Units}{old}
```

now `\LenUnit`, `\MasUnit` will expand as yard, pound respectively.

Notice that if the specified group already exists then `\AddCMDToGroup` just appends the speficied commands to it. In particular this

```
\AddCMDToGroup{Units}{\LenUnit}
\AddCMDToGroup{Units}{\MasUnit}
```

is equivalent to write

```
\AddCMDToGroup{Units}{\LenUnit, \MasUnit}
```

7 Advanced usage

This section is for advanced users and package mantainers that knows LaTeX3, It's not needed to use `styledcmd` daily or creating documents. If `\newstyledcmd`, `\NewDocStyledCMD` and `\newstyledcmdExp` aren't suitable for you it's possible to create a custom styled command generator, but we first need to know a bit of the internal structure of `styledcmd`.

What you see as a styled command it's instead a collection of different macros:

- multiple *effective styled commands* (*ES commands*), one for each style;
- a single *dispatch command* that's called by the user and expands to the specified ES command.

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

`\stycmd_generate:NNN` ⟨*generator name*⟩ ⟨*ES commands generator*⟩ ⟨*dispatch command generator*⟩
`\stycmd_generate:NN` ⟨*generator name*⟩ ⟨*ES commands generator*⟩

Creates a generator of styled commands with name ⟨*generator name*⟩. Argument ⟨*ES commands generator*⟩ is used to create ES commands and should accept a macro name as the first argument, but there aren't other restrictions on remaining arguments. Suitable ES commands generators are `\newcommand` and `\NewDocumentCommand`.

Argument ⟨*dispatch command generator*⟩ should generate the dispatch command. Despite ⟨*ES commands generator*⟩ this command must have only one parameter, a string representing the command to be created. Suitable values for this parameter are:

`\stycmd_xparsecmd:N`

`\stycmd_xparsecmd:N` ⟨*command*⟩

Creates the dispatch command with `\ProvideDocumentCommand` with optional style name parameter (used in `\newstyledcmd` and `\NewDocStyledCMD`).

`\stycmd_expcmd:N`

`\stycmd_expcmd:N` ⟨*command*⟩

Creates the dispatch command with `\providecommand` with mandatory style name parameter (used in `\newstyledcmdExp`).

If you don't specify the dispatch command generator (by using the `NN` variant) `\stycmd_xparsecmd:N` is used implicitly.

`\stycmd_generate_renew:NN` `\stycmd_generate_renew:NN` ⟨*updater name*⟩ ⟨*ES commands generator*⟩

Creates command `\`⟨*updater name*⟩ that modifies styles of commands already generated, like `\renewcommmand` edits commands generated by `\newcommand`.

For example if you have `\createA` that creates commands and `\editA` that edits commands generated by `\createA` then

```
\stycmd_generate:NN \createStyledA \createA
```

creates styled commands by using `\createA` and

```
\stycmd_generate_renew:NN \editStyledA \editA
```

modifies commands generated by `\createStyledA` (by invoking `\editA`).

8 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:nn { stycmd }{ proxy }
\proxy_push_member:Vnn \c__stycmd_cmdproxy_str { default }{ tl }

```

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

`\__stycmd_cmd:n` Entity name

```

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

```

(*End definition for* `\__stycmd_cmd:n`.)

`\__stycmd_setdef_strip:Nn` Changes the default style

```

\cs_new_protected:Nn \__stycmd_setdef_aux:nN
  {
    \object_member_set:nnn
      {
        \__stycmd_cmd:n{ #1 }
      }
      { default }
      { #2 }
  }
\cs_generate_variant:Nn \__stycmd_setdef_aux:nN { nc }

```

```
\cs_new_protected:Nn \__stycmd_setdef_style:nn
  {
    \__stycmd_setdef_aux:nc{ #1 }
      {
        \object_macro_adr:nn
          {
            \__stycmd_cmd:n{ #1 }
          }
          {
            style - #2
          }
      }
  }

\cs_generate_variant:Nn \__stycmd_setdef_style:nn { ff }

\cs_new_protected:Nn \__stycmd_setdef_strip:Nn
  {
    \__stycmd_setdef_style:ff{ \cs_to_str:N #1 }
      { \tl_trim_spaces:n{ #2 } }
  }

```

(*End definition for* `\__stycmd_setdef_strip:Nn`*.*)

`\__stycmd_cmd_define_strip:NNn` Define a macro with the specified command

```

\cs_new_protected:Nn \__stycmd_cmd_define_aux_aux:NN
  {
    #1 { #2 }
  }
\cs_generate_variant:Nn \__stycmd_cmd_define_aux_aux:NN { Nc }

\cs_new_protected:Nn \__stycmd_cmd_define_aux:Nnn
  {
    \__stycmd_cmd_define_aux_aux:Nc #1
      {
        \object_macro_adr:nn
          {
            \__stycmd_cmd:n{ #2 }
          }
          {
            style - #3
          }
      }
  }

\cs_generate_variant:Nn \__stycmd_cmd_define_aux:Nnn { Nff }

\cs_new_protected:Nn \__stycmd_cmd_define_strip:NNn
  {
    \__stycmd_cmd_define_aux:Nff #1
      { \cs_to_str:N #2 }{ \tl_trim_spaces:n { #3 } }
  }

```

(*End definition for* `\__stycmd_cmd_define_strip:NNn`.)

`\__stycmd_cmd_usedef:N`
`\__stycmd_cmd_usesty_strip:Nn`

Uses the styled command

```

\cs_new:Nn \__stycmd_cmd_usedef_aux:n
  {
    \object_member_use:nn
      {
        \__stycmd_cmd:n{ #1 }
      }
      {
        default
      }
  }

\cs_generate_variant:Nn \__stycmd_cmd_usedef_aux:n { f }

\cs_new:Nn \__stycmd_cmd_usedef:N
  {
    \__stycmd_cmd_usedef_aux:f{ \cs_to_str:N #1 }
  }


\cs_new:Nn \__stycmd_cmd_usesty_aux:nn
  {
    \object_macro_use:nn
      {
        \__stycmd_cmd:n{ #1 }
      }
      {
        style - #2
      }
  }

\cs_generate_variant:Nn \__stycmd_cmd_usesty_aux:nn { ff }

\cs_new:Nn \__stycmd_cmd_usesty_strip:Nn
  {
    \__stycmd_cmd_usesty_aux:ff{ \cs_to_str:N #1 }
      { \tl_trim_spaces:n{ #2 } }
  }

```

(*End definition for* `\__stycmd_cmd_usedef:N` *and* `\__stycmd_cmd_usesty_strip:Nn`.)

`\__stycmd_entity_create_strip:Nnn`

Creates a new entity if it doesn't exists and execute following code

```

\cs_new_protected:Nn \__stycmd_entity_create_aux:nnn
  {
    \object_if_exist:nF
      {
        \__stycmd_cmd:n{ #1 }
      }
      {
        \object_create:Vnn \c__stycmd_cmdproxy_str
          { stycmd }{ entity - #1 }

        \__stycmd_setdef_style:nn{ #1 }{ #2 }

        #3
      }
  }

\cs_generate_variant:Nn \__stycmd_entity_create_aux:nnn { ffn }

\cs_new_protected:Nn \__stycmd_entity_create_strip:Nnn
  {
    \__stycmd_entity_create_aux:ffn
      { \cs_to_str:N #1 }
      { \tl_trim_spaces:n{ #2 } }
      { #3 }
  }

```

(*End definition for* `\__stycmd_entity_create_strip:Nnn`*.*)

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

```

\cs_new_protected:Nn \stycmd_xparsecmd:N
  {
    \ProvideDocumentCommand { #1 } { o }
      {
        \IfNoValueTF {##1}
          {
            \__stycmd_cmd_usedef:N #1
          }
          {
            \__stycmd_cmd_usesty_strip:Nn #1 { ##1 }
          }
      }
  }

```

(*End definition for* `\stycmd_xparsecmd:N`*. This function is documented on page 4.*)

`\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:N
  {
    \providecommand { #1 } [1]
      {
        \tl_if_empty:nTF {##1}
          {
            \__stycmd_cmd_usedef:N #1
          }
          {
            \__stycmd_cmd_usesty_strip:Nn #1 { ##1 }
          }
      }
  }

```

(*End definition for* `\stycmd_expcmd:N`*. This function is documented on page 4.*)

`\SetGlobalStyle`

Change the default style for specified command

```

\NewDocumentCommand{\SetGlobalStyle}{m m}
  {
    \__stycmd_setdef_strip:Nn #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_protected:Nn \stycmd_generate:NNN
  {
    \cs_new_protected:Npn #1 ##1 ##2
      {
        \__stycmd_entity_create_strip:Nnn ##1 { ##2 }
          {
            #3 ##1
          }
        \__stycmd_cmd_define_strip:NNn #2 ##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_cmd_define_strip:NNn #2 ##1 { ##2 }
      }
  }

```

(*End definition for* `\stycmd_generate:NNN`*,* `\stycmd_generate:NN`*, and* `\stycmd_generate_renew:NN`*. These functions are documented on page 4.*)

`\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.*)

`\newstyledcmdExp`
`\renewstyledcmdExp`
`\providestyledcmdExp`

Declare a new macro with the specified style name.

```
\stycmd_generate:NNN \newstyledcmdExp \newcommand \stycmd_expcmd:N
\stycmd_generate_renew:NN \renewstyledcmdExp \renewcommand
\stycmd_generate:NNN \providestyledcmdExp \providecommand \stycmd_expcmd:N
```

(*End definition for* `\newstyledcmdExp`*,* `\renewstyledcmdExp`*, and* `\providestyledcmdExp`*. These functions are documented on page 3.*)

`\AddCMDToGroup`

Creates a group of commands

```

\str_new:N \g__stycmd_grproxy_str
\seq_new:N \g__stycmd_tmp_seq
\seq_new:N \g__stycmd_tmpb_seq

\proxy_create_gset:Nnn \g__stycmd_grproxy_str { stycmd }{ groups }

\proxy_push_member:Vnn \g__stycmd_grproxy_str { commands }{ seq }

\cs_generate_variant:Nn \seq_gconcat:NNN { ccN }

\cs_new_protected:Nn \__stycmd_gconcat:nN
  {
    \seq_gconcat:ccN { #1 }{ #1 } #2
  }

\cs_new_protected:Nn \__stycmd_addgroup:nn
  {
    \object_if_exist:nF
      {
        \object_address:nn{ stycmd }{ group - #1 }
      }
      {
        \object_create:Vnn \g__stycmd_grproxy_str
          { stycmd }
          { group - #1 }
      }

    \seq_gset_from_clist:Nn \g__stycmd_tmp_seq { #2 }
    \seq_gset_map_x:NNn \g__stycmd_tmpb_seq \g__stycmd_tmp_seq
      {
        \cs_to_str:N ##1
      }

    \__stycmd_gconcat:nN
```

```
      {
      \object_member_adr:nnn
        {
          \object_address:nn
            { stycmd }
            { group - #1 }
        }
        { commands }
        { seq }
      } \g__stycmd_tmpb_seq
  }

\cs_generate_variant:Nn \__stycmd_addgroup:nn { fn }

\NewDocumentCommand{\AddCMDToGroup}{m m}
  {
    \__stycmd_addgroup:fn { \tl_trim_spaces:n{ #1 } } { #2 }
  }

```

(*End definition for* `\AddCMDToGroup`*. This function is documented on page 3.*)

`\SetGroupStyle` Change the default style for each command in the group.

```
\NewDocumentCommand{\SetGroupStyle}{m m}
  {
    \seq_map_inline:cn
      {
        \object_member_adr:nnn
          {
            \object_address:nn
              { stycmd }
              { group - #1 }
          }
          { commands }
          { seq }
      }
      {
        \__stycmd_setdef_style:nn{ ##1 }{ #2 }
      }
  }
```

(*End definition for* `\SetGroupStyle`*. This function is documented on page 3.*)

Legacy commands

```

\msg_new:nnnn{ styledcmd }{ stydep }{ Old-fashioned~groups~for~styled~commands~are~deprecate
  {
    From~version~2.0~of~styledcmd~you~should~use~new~group~commands~instea~of~old~ones,~see~
  }

\str_new:N \g__stycmd_act_group_str
\str_new:N \g__stycmd_act_style_str

\NewDocumentCommand{\styBeginGroup}{ m }
  {
    \msg_warning:nn{ styledcmd }{ stydep }
    \str_gset:Nn \g__stycmd_act_group_str{ #1 }
  }

\NewDocumentCommand{\styEndGroup}{}
  {
    \msg_warning:nn{ styledcmd }{ stydep }
    \str_gset:Nn \g__stycmd_act_group_str{}
  }

\NewDocumentCommand{\styBeginStyle}{ m }
  {
    \msg_warning:nn{ styledcmd }{ stydep }
    \str_gset:Nn \g__stycmd_act_style_str{ #1 }
  }

\NewDocumentCommand{\styEndStyle}{}
  {
    \msg_warning:nn{ styledcmd }{ stydep }
    \str_gset:Nn \g__stycmd_act_style_str{}
  }

\cs_generate_variant:Nn \__stycmd_addgroup:nn { Vn }

\NewDocumentCommand{\newGstyledcmd}{m o m}
  {
    \msg_warning:nn{ styledcmd }{ stydep }
    \__stycmd_addgroup:Vn \g__stycmd_act_group_str { #1 }
    \IfNoValueTF{ #2 }
      {
        \exp_args:NNV \newstyledcmd #1 \g__stycmd_act_style_str { #3 }
      }
      {
        \exp_args:NNV \newstyledcmd #1 \g__stycmd_act_style_str [ #2 ] { #3 }
      }
  }

\NewDocumentCommand{\NewGDocStyledCMD}{m m m}
  {
    \msg_warning:nn{ styledcmd }{ stydep }
    \__stycmd_addgroup:Vn \g__stycmd_act_group_str { #1 }
    \exp_args:NNV \NewDocStyledCMD #1 \g__stycmd_act_style_str { #2 } { #3 }
  }

 \NewDocumentCommand{\setGroupStyle}{m m}
   {
     \msg_warning:nn{ styledcmd }{ stydep }
     \SetGroupStyle{ #1 }{ #2 }
   }
```

⟨/package⟩