

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

<code>\newstyledcmd</code>	<code>\newstyledcmd {\langle macro name \rangle} {\langle style name \rangle} [\langle number of arguments \rangle] {\langle code \rangle}</code>
----------------------------	---

<code>\renewstyledcmd</code>
<code>\providestyledcmd</code>

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

The most important feature is that you can call `\newstyledcmd` multiple times with the same $\langle macro name \rangle$ but different $\langle style name \rangle$, in this way each of $\langle macro name \rangle [\langle style name \rangle]$ expands to $\langle code \rangle$ associated to specified $\langle style name \rangle$. 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}
\sauto[informal]{uncle}
\sauto[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 explicitly 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 \NewDocStyledCMD {\command name} {\format name} {\arguments format} {\code}
\RenewDocStyledCMD
\ProvideDocStyledCMD
```

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

Commands 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

<code>\newstyledcmdExp</code>	<code>\newstyledcmdExp {\macro name} {\style name} [(number of arguments)] {\code}</code>
<code>\renewstyledcmdExp</code>	
<code>\providestyledcmdExp</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 together their style. Groups are created with the following command

<code>\NewStyledGroup</code>	<code>\NewStyledGroup {\group name} {\comma list of commands}</code>
------------------------------	--

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

<code>\SetGroupStyle</code>	<code>\SetGroupStyle {\group name} {\style name}</code>
-----------------------------	---

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

```
\NewStyledGroup{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.

7 Advanced usage

This section is for advanced users and package maintainers that knows L^AT_EX3, 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.

<code>\stycmd_generate:NNN</code>	<code>\stycmd_generate:NNN <generator name> <ES commands generator> <dispatch command generator></code>
<code>\stycmd_generate:NN</code>	<code>\stycmd_generate:NN <generator name> <ES commands generator></code>

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:

<code>\stycmd_xparsecmd:n</code>	<code>\stycmd_xparsecmd:n {(command name string)}</code>
----------------------------------	--

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

<code>\stycmd_expcmd:n</code>	<code>\stycmd_expcmd:n {(command name string)}</code>
-------------------------------	---

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.

<code>\stycmd_generate_renew:NN</code>	<code>\stycmd_generate_renew:NN <updater name> <ES commands generator></code>
--	---

Creates command `\<updater name>` that modifies styles of commands already generated, like `\renewcommand` 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

```

1 < *package>
2 < @@=stycmd>

\c__stycmd_cmdproxy_str Proxy used to generate styled commands
3 \str_const:Nx \c__stycmd_cmdproxy_str { \object_address:nn
4   { stycmd }{ proxy } }
5
6 \proxy_create:nnN { stycmd }{ proxy } \c_object_public_str
7 \proxy_push_member:Vnn \c__stycmd_cmdproxy_str { default }{ tl }
8

(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
9 \cs_new:Nn \__stycmd_cmd:n
10 {
11   \object_address:nn{ stycmd }{ entity - #1 }
12 }
13
14 \cs_new:Nn \__stycmd_cmd_style:nn
15 {
16   \object_member_adr:nnn{ \__stycmd_cmd:n{ #1 } }{ style - #2 }
17   { stycmd_void }
18 }
19
20 \cs_new:Nn \stycmd_void_use:N { #1 }
21 \cs_new_eq:NN \stycmd_void_use:c \use:c
22
23 \cs_new:Nn \__stycmd_cmd_default:n
24 {
25   \object_member_adr:nn{ \__stycmd_cmd:n{ #1 } }{ default }
26 }
27

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

\stycmd_xparsecmd:n Defines the main macro with \ProvideDocumentCommand.
28
29 \cs_new_protected:Nn \__stycmd_xparsecmd_aux:Nn
30 {
31   \ProvideDocumentCommand { #1 } { o }
32   {
33     \IfNoValueTF {##1}
34     {
35       \object_member_use:nn
36       {
37         \__stycmd_cmd:n{ #2 }
38       }
39       {
40         default
41       }
42     }

```

```

43     {
44         \object_member_use:nnn
45         {
46             \__stycmd_cmd:n{ #2 }
47         }
48         {
49             style - ##1
50         }
51         { stycmd_void }
52     }
53 }
54 }
55
56 \cs_generate_variant:Nn \__stycmd_xparsecmd_aux:Nn { cn }
57
58 \cs_new_protected:Nn \stycmd_xparsecmd:n
59 {
60     \__stycmd_xparsecmd_aux:cn { #1 }{ #1 }
61 }
62

```

(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.

```

63
64 \cs_new_protected:Nn \__stycmd_expcmd_aux:Nn
65 {
66     \providecommand { #1 } [1]
67     {
68         \tl_if_empty:nTF {##1}
69         {
70             \object_member_use:nn
71             {
72                 \__stycmd_cmd:n{ #2 }
73             }
74             {
75                 default
76             }
77         }
78         {
79             \object_member_use:nnn
80             {
81                 \__stycmd_cmd:n{ #2 }
82             }
83             {
84                 style - ##1
85             }
86             { stycmd_void }
87         }
88     }
89 }
90

```

```

91 \cs_generate_variant:Nn \__stycmd_expcmd_aux:Nn { cn }
92
93 \cs_new_protected:Nn \stycmd_expcmd:n
94 {
95   \__stycmd_expcmd_aux:cn { #1 }{ #1 }
96 }
97

```

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

`\setGlobalStyle` Change the default style for specified command

```

98
99 \cs_new_protected:Nn \__stycmd_setdef:nN
100 {
101
102   \object_member_set:nnn
103   {
104     \__stycmd_cmd:n{ #1 }
105   }
106   { default }
107   { #2 }
108 }
109 \cs_generate_variant:Nn \__stycmd_setdef:nN { nc }
110
111 \cs_new_protected:Nn \__stycmd_setdef_style:nn
112 {
113   \__stycmd_setdef:nc{ #1 }
114   {
115     \__stycmd_cmd_style:nn{ #1 }{ #2 }
116   }
117 }
118
119 \cs_generate_variant:Nn \__stycmd_setdef_style:nn { fn }
120 \cs_new_protected:Nn \__stycmd_chdef:Nn
121 {
122   \__stycmd_setdef_style:fn{ \cs_to_str:N #1 }{ #2 }
123 }
124
125 \NewDocumentCommand{\setGlobalStyle}{m m}
126 {
127   \__stycmd_chdef:Nn #1 { #2 }
128 }
129

```

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

`\stycmd_generate:NNN` Declare the styled version #1 of the macro generator command #2. the `_renew` variant
`\stycmd_generate:NN` requires a preceding declaration
`\stycmd_generate_renew:NN`

```

130
131 \cs_generate_variant:Nn \__stycmd_pars:NN { cc }
132
133 \cs_new_protected:Nn \__stycmd_generate_aux:NNnn
134 {
135   \object_if_exist:nF

```

```

136     {
137         \__stycmd_cmd:n{ #3 }
138     }
139     {
140         \object_create:VnnNN \c__stycmd_cmdproxy_str
141         { stycmd }{ entity - #3 }
142         \c_object_global_str
143         \c_object_public_str
144
145         \__stycmd_setdef_style:nn{ #3 }{ #4 }
146
147         #2 { #3 }
148     }
149     \exp_args:Nc #1
150     {
151         \__stycmd_cmd_style:nn{ #3 }{ #4 }
152     }
153
154 }
155
156 \cs_generate_variant:Nn \__stycmd_generate_aux:NNnn { NNfn }
157
158 \cs_new_protected:Nn \__stycmd_generate_aux_cmd:NNNn
159 {
160     \__stycmd_generate_aux:NNfn #1 #2 { \cs_to_str:N #3 }{ #4 }
161 }
162
163 \cs_new_protected:Nn \__stycmd_generate_renew_aux:Nnn
164 {
165     \exp_args:Nc #1
166     {
167         \__stycmd_cmd_style:nn{ #2 }{ #3 }
168     }
169
170 }
171
172 \cs_new_protected:Nn \stycmd_generate:NNN
173 {
174     \cs_new_protected:Npn #1 ##1 ##2
175     {
176         \__stycmd_generate_aux_cmd:NNNn #2 #3 ##1 { ##2 }
177     }
178 }
179 \cs_new_protected:Nn \stycmd_generate:NN
180 {
181     \stycmd_generate:NNN #1 #2 \stycmd_xparsecmd:n
182 }
183
184
185 \cs_new_protected:Nn \stycmd_generate_renew:NN
186 {
187     \cs_new_protected:Npn #1 ##1 ##2
188     {
189         \__stycmd_generate_renew_aux:Nnn #2 { ##1 }{ ##2 }

```



```

190     }
191 }
192

```

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

```

\newstyledcmd Declare a new macro with the specified style name.
\renewstyledcmd 193 \stycmd_generate:NN \newstyledcmd \newcommand
\providestyledcmd 194 \stycmd_generate_renew:NN \renewstyledcmd \renewcommand
195 \stycmd_generate:NN \providestyledcmd \providecommand

```

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

```

\NewDocStyledCMD Declare a new styled macro with the \NewDocumentCommand syntax.
\RenewDocStyledCMD 196 \stycmd_generate:NN \NewDocStyledCMD \NewDocumentCommand
\ProvideDocStyledCMD 197 \stycmd_generate_renew:NN \RenewDocStyledCMD \RenewDocumentCommand
198 \stycmd_generate:NN \ProvideDocStyledCMD \ProvideDocumentCommand

```

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

```

\newstyledcmdExp Declare a new macro with the specified style name.
\renewstyledcmdExp 199 \stycmd_generate:NNN \newstyledcmdExp \newcommand \stycmd_expcmd:n
\providestyledcmdExp 200 \stycmd_generate_renew:NN \renewstyledcmdExp \renewcommand
201 \stycmd_generate:NNN \providestyledcmdExp \providecommand \stycmd_expcmd:n

```

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

\NewStyledGroup Creates a group of commands

```

202
203 \str_new:N \g__stycmd_grproxy_str
204 \seq_new:N \g__stycmd_tmp_seq
205 \seq_new:N \g__stycmd_tmpb_seq
206
207 \proxy_create_gset:NnnN \g__stycmd_grproxy_str { stycmd }{ groups }
208   \c_object_public_str
209
210 \proxy_push_member:Vnn \g__stycmd_grproxy_str { commands }{ seq }
211
212 \cs_new_protected:Nn \__stycmd_newgroup:nn
213 {
214   \object_create:VnnNN \g__stycmd_grproxy_str
215     { stycmd }
216     { group - #1 }
217     \c_object_global_str
218     \c_object_public_str
219
220   \seq_gset_from_clist:Nn \g__stycmd_tmp_seq { #2 }
221   \seq_gset_map_x:Nnn \g__stycmd_tmpb_seq \g__stycmd_tmp_seq
222     {
223       \cs_to_str:N ##1
224     }

```

```

225
226 \object_member_set_eq:nnnN
227 {
228   \object_address:nn
229   { stycmd }
230   { group - #1 }
231 }
232 { commands }
233 { seq }
234 \g__stycmd_tmpb_seq
235 }
236
237 \NewDocumentCommand{\NewStyledGroup}{m m}
238 {
239   \__stycmd_newgroup:nn { #1 } { #2 }
240 }
241

```

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

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

```

242 \NewDocumentCommand{\SetGroupStyle}{m m}
243 {
244   \seq_map_inline:cn
245   {
246     \object_member_adr:nnn
247     {
248       \object_address:nn
249       { stycmd }
250       { group - #1 }
251     }
252     { commands }
253     { seq }
254   }
255   {
256     \__stycmd_setdef_style:nn{ ##1 }{ #2 }
257   }
258 }

```

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

```

259 \endpackage

```