

The ccool package*

Erwann Rogard[†]

Released 2021-09-20

Abstract

This \LaTeX package provides an interface to define and evaluate key-based replacement rules[3]. It can be used to parse the argument specification of a document command[4].

Contents

I	Usage	3
0.1	Core feature	3
0.2	Process the <i>val_i</i> 's	3
0.3	Append to a hook	3
0.4	Expand the <i>val_i</i> 's	3
0.5	Head	4
0.6	Tail	4
0.7	Parameterize the <i>key_i</i> 's	4
0.8	Write	4
II	Other	5
1	Bibliography	5
2	Do's and dont's	6
3	To do	6
4	Support	6
III	Listing	7
	Tutorial >	7
	1. Let \mathbb{N} and \mathbb{R} denote...	7

*This file describes version v3.2, last revised 2021-09-20.

[†]first.lastname at gmail.com

2. Same as 1, with <code>\NewDocumentCommand</code>	7
3. Same as 2, with <code>\Ccool</code>	7
4. Same as 3, with expansion	7
5. Same as 3, parameterized	7
<	7
6. Mittelwertsatz für n Variable.	7
7. Listing 6 read from file	8
8. Probability space	8
9. Families of polynomial functions	9
10. Listing 9 read from file	9
12. Listing 11 read from file	9
13. CUSUM statistic.	10
14. Listing 13 read from file	11
IV Implementation	11
1 Opening	11
2 aux	11
3 lang	13
4 log	15
5 make_key	16
6 make_ccool	17
7 msg	18
8 option	18
9 prop	19
10 seq	21
11 seq_use	21
12 Front-end	22
13 Closing	25

Part I

Usage

`\Ccool[usage:cs:ccool]`

`\Ccool` [$\langle t_1 \rangle$] $\langle t_2 \rangle$ $c\langle code_1 \rangle\{keyval list_1\}+*s\langle separators \rangle c\langle code_2 \rangle$ [$\langle t_6 \rangle$]
 where $\langle separators \rangle$ is either of: $\{tl_3\}$, $\{tl_3\}\{tl_4\}$, and $\{tl_3\}\{tl_4\}\{tl_5\}$.

Semantics See [subsection 0.1-0.8](#).

0.1 Core feature

`\Ccool\{keyval list_1\}` executes for each $\langle key_i \rangle = \langle val_i \rangle$,

- 1) $\langle val_i \rangle \leftarrow \text{\function}\{\langle val_i \rangle\}$
- 2) define $\langle key_i \rangle$ such that $\langle key_i \rangle \rightarrow \langle val_i \rangle$,

where `\function` is encoded in *global option Inner*. For instance, the side effect of `\Ccool{ Real = \mathbb{R} }` is $\text{\Real} \rightarrow \text{\mathbb{R}}$. To be sparingly used, *global option Expans* controls the type of expansion of $\langle key_i \rangle$ and $\langle val_i \rangle$.

if $\langle key_i \rangle$ needs arguments, use `\lambdax` from `lambdax` on the rhs.

0.2 Process the val_i 's

`\Ccool c\langle code_1 \rangle\{keyval list_1\}` is identical to the **Core feature**, except it overrides *Inner*.

In our example, if multiple number systems are defined with `\Ccool` (natural, reals, ...), it is more efficient to omit `\mathbb{.}` inside $\langle val_i \rangle$, and instead use `c\{\mathbb{#1}\}`, where `#1` means "parameter to be replaced".

0.3 Append to a hook

`\Ccool\{keyval list_1\}+` is identical to the **Core feature**, except it repeats after `\CcoolHook`. This is useful to make the side effect persist after a *local group* (such as `theorem`).

0.4 Expand the val_i 's

`\Ccool\{keyval list_1\}*` supplements the **Core feature** with the expansion of the $\langle val_i \rangle$'s using typesetting rules encoded in *global option Separand Outer*. The first are *separators* applied to the $\langle val_i \rangle$'s to form a *token list*, and the second a function applied to the latter.

They can be overridden inline by appending further `s\langle separators \rangle` and `c\langle code_2 \rangle`, respectively, to the list of arguments.

0.5 Head

`\Ccool[⟨tl1⟩]{⟨keyval list1⟩}` expands ⟨tl₁⟩ and executes the **Core feature**.

There may be situations where it is convenient to pass ⟨tl₁⟩ as empty.

0.6 Tail

`\Ccool{⟨keyval list1⟩}[⟨tl6⟩]{⟨keyval list2⟩}` is identical to `\Ccool{⟨keyval list1⟩}` followed by `\Ccool[⟨tl6⟩]{⟨keyval list2⟩}`.

The combination of **Core feature**, **Head**, and **Tail** allows to integrate typesetting and the creation of commands.

0.7 Parameterize the key_i's

`\Ccool⟨tl2⟩{⟨keyval list1⟩}` is identical to the **Core feature**, except ⟨key_i⟩ is replaced by ⟨key_i<tl₂>⟩. The default value of ⟨tl₂⟩ is encoded in **Param**. In our example, ⟨tl₂⟩ could be **Style**.

0.8 Write

global option **Write** is identical to the **Core feature**, except that if **Write** is set to `\BooleanTrue`, the code is written to a file whose path is encoded in *global option* **File**.

`\CcoolClearusage:cs:clear`

`\CcoolClear` `\CcoolClear⟨tl2⟩{...⟨keyi⟩,...}`

Semantics Clears all `\⟨keyi<tl2>⟩`'s

`\CcoolHookusage:cs:hook`

`\CcoolHook` `\CcoolHook`

Semantics No side effect or expansion

`\CcoolOptionusage:cs:option`

`\CcoolOption` `\CcoolOption[...⟨keyi⟩|⟨keyi⟩=⟨vali⟩,...]`

where ⟨key_i⟩ is either of **And**, **Expans**, **File**, **Inner**, **Param**, **Outer**, **Separ**, and **Write**.

Semantics Modify the default behavior of `\Ccool`

And `Andusage:opt:an`

Semantics Sets the translation of *and* in language ⟨key⟩ to ⟨val⟩

Syntax ⟨keyval list⟩

Expans `Expansusage:opt:ex`

Syntax `eo|ee|ex|xo|xe|xx`

File `Fileusage:opt:fi`

Syntax $\langle path \rangle$

Inner Innerusage:opt:in

Syntax $\langle code \rangle$, with `####1` as the *placeholder*

Param Paramusage:opt:pa

Syntax $\langle token list \rangle$

Outer Outerusage:opt:ou

Default `\ensuremath{####1}`

Syntax $\langle code \rangle$, with `####1` as the *placeholder*

Separ Separusage:opt:se

Other Default behavior depends on whether `babel` and `amsmath` are loaded

Syntax That of *separators* in [3, Section 8 of l3seq]

Write Writeusage:opt:wr

Syntax `\BooleanFalse|\BooleanTrue`

`\CcoolReadusage:cs:read`

`\CcoolRead` `\CcoolRead[\langle path \rangle]`

Semantics

1. Reads the definitions in $\langle path \rangle$.
2. Writes to `ccool.log`: ‘read from $\langle path \rangle$ ’

`\CcoolVersusage:cs:vers`

`\CcoolVers` `\CcoolVers`

Semantics \rightarrow the package’s version

Part II

Other

1 Bibliography

- [1] Olympia Hadjiliadis. “Change–point detection of two–sided alternatives in the Brownian motion model and its connection to the gambler’s ruin problem with relative wealth perception”. PhD thesis. Columbia University, 2005.
- [2] Thomas F. Sturm. *The tcolorbox package*. <http://www.texdoc.net/texmf-dist/doc/latex/tcolorbox/tcolorbox.pdf>. 2019.

- [3] The L^AT_EX3 Project Team. *The L^AT_EX3 interfaces*. <https://ctan.math.washington.edu/tex-archive/macros/latex/contrib/l3kernel/expl3.pdf>. 2019.
- [4] The L^AT_EX3 Project Team. *The xparse package*. <https://ctan.math.illinois.edu/macros/latex/contrib/l3packages/xparse.pdf>. 2019.
- [5] @frougon. “*Journaling calls to a function []*”. <https://tex.stackexchange.com/a/536620>. 2020.
- [6] @Javier Bezos. *When loading babel with spanish, spurious document command parser*. <https://tex.stackexchange.com/a/547018/112708>. 2020.

2 Do’s and dont’s

1.

Don’t: `Inner=\{####1\}`

Symptom: `\CcoolRead` fails

Do: `Inner={\char‘{####1\char‘}}`

2.

Don’t: `\langle key_i \rangle x$`.

Do: `\langle key_i \rangle {<} x$`

3.

Don’t: `[a, b)`

Do: `{[]a, b{)}`

4.

Don’t: `\cal F`.

Do: `\cal{F}` or `\mathcal{F}`

5.

Don’t: `\[x_0, x\]`

Do: `\left[x_0, x\right]`

6.

Don’t: `\usepackage[spanish]{babel}`

Do: `\usepackage[spanish.noquoting]{babel}[6]`

3 To do

- 1. Create an environment for `\CoolHook`.

4 Support

This package is available from <https://github.com/rogard/ccool>.

Part III

Listing

Listing 1. “Let \mathbb{N} and \mathbb{R} denote...”

```
Let~$\mathbb{N}$ and $\mathbb{R}$ denote the natural and real numbers.
```

Let \mathbb{N} and \mathbb{R} denote the natural and real numbers.

Listing 2. Same as 1, with `\NewDocumentCommand`

```
\DeclareDocumentCommand\Nat{}{\mathbb{N}}
\DeclareDocumentCommand\Real{}{\mathbb{R}}
Let~$\Nat$ and $\Real$ denote the natural and real numbers.
```

Let \mathbb{N} and \mathbb{R} denote the natural and real numbers.

Listing 3. Same as 2, with `\Ccool`

```
\begingroup
\Ccool c{\mathbb{#1}}{ Nat = {N}, Real = {R} }
Let~$\Nat$ and $\Real$~denote the natural and real numbers.
\endgroup
```

Let \mathbb{N} and \mathbb{R} denote the natural and real numbers.

Listing 4. Same as 3, with expansion

```
\begingroup
\Ccool[Let~]
c{\mathbb{#1}}{ Nat = {N}, Real = {R} }*
[~denote the natural and real numbers.]{~}
\endgroup
```

Let \mathbb{N} and \mathbb{R} denote the natural and real numbers.

Listing 5. Same as 3, parameterized

```
\begingroup
\Ccool<foo>c{\mathbb{#1}}{ Nat = {N}, Real = {R} }
[Let ~$\Nat<foo>$ and ~$\Real<foo>$ denote the natural and real numbers.]{~}
\endgroup
```

Let \mathbb{N} and \mathbb{R} denote the natural and real numbers.

Listing 6. Mittelwertsatz für n Variable [2, p. 17.3]

```

\begingroup
\CoolOption[ Write = \BooleanTrue ]
\selectlanguage{german}
\newtheorem{theorem}{Theorem}
\AfterEndEnvironment{theorem}{\CoolHook}
\Cool c{\mathbb{#1}}
{ N = { N } , R = { R } }+[]
{ Grad = { \operatorname{grad} } }+
[\begin{theorem}
[Mittelwertsatz f\"ur $n$ Variable]Es~sei~]
{ OffMenge = {D}, Ci = {C^{1}}, Strecke = { \left[x_0,x\right] } }+
[$n\in\mathbb{N}$,~$\text{OffMenge}\subseteq\mathbb{N}^n$ eine offene Menge und
$f\in C^1(\text{OffMenge},\mathbb{R})$].
Dann gibt es auf jeder Strecke $Strecke\subseteq\text{OffMenge}$ einen Punkt
$\xi\in\text{Strecke}$,~]
{ Steig = { \frac{ f(x)-f(x_0) }{ x-x_0 } }, Punkt = { \xi } }+
[so dass gilt
\begin{equation*}
\text{Steig} = \text{Grad } f(\text{Punkt})^{\top}
\end{equation*}
\end{theorem}]
{}
(Check: $N$, $\xi$)
\endgroup
\CoolOption

```

Theorem 1 (Mittelwertsatz für n Variable) *Es sei $n \in \mathbb{N}$, $D \subseteq \mathbb{N}^n$ eine offene Menge und $f \in C^1(D, \mathbb{R})$. Dann gibt es auf jeder Strecke $[x_0, x] \subset D$ einen Punkt $\xi \in [x_0, x]$, so dass gilt*

$$\frac{f(x) - f(x_0)}{x - x_0} = \text{grad}f(\xi)^\top$$

(Check: \mathbb{N} , ξ)

Listing 7. Listing 6 read from file

```

\CoolRead ~$N$ $R$ $\text{OffMenge}$ $Ci$ $\text{Strecke}$
\CoolClear

```

$\mathbb{N} \ \mathbb{R} \ D \ C^1 [x_0, x]$

Listing 8. Probability space

```

\begingroup
\Cool[Let~]
{ Space = \Omega, Field = \mathcal{F}, Meas = \mathcal{P} }
*s{\{,\}}c{\#\}

```



```
[~denote the probability space, where~]{ PowerSet = { 2^{\Space} } }
[{\Field\subset \PowerSet$.]
{}
\endgroup
```

Let $\{\Omega, \mathcal{F}, \mathcal{P}\}$ denote the probability space, where $\mathcal{F} \subset 2^\Omega$.

Listing 9. Families of polynomial functions

```
\CcoolOption[ Write = \BooleanTrue ]
\Ccool c{\mathbb{#1}}{ Nat = {N}, Real = {R} }
[Let~]
{ PolyR = \LambdaX{o}{\Real\IfValueT{#1}{_#1}{X} } }
[{\PolyR[n]} and {\PolyR}, denote the families of polynomial functions
  on {\Real}, of order $n$ et and their union over $n \in \Nat$,
  respectively. ]
{}
\CcoolClear
\CcoolOption
```

Let $\mathbb{R}_n[X]$ and $\mathbb{R}[X]$, denote the families of polynomial functions on \mathbb{R} , of order n et and their union over $n \in \mathbb{N}$, respectively.

Listing 10. 9 read from file

```
\begingroup
\CcoolRead {\PolyR[n]} et {\PolyR}
\endgroup
```

$\mathbb{R}_n[X]$ et $\mathbb{R}[X]$

Listing 11. Fonction et fonctionnelle

```
\CcoolOption[ Write = \BooleanTrue ]
\selectlanguage{french}
\Ccool<eval>{ fun = \LambdaX{(#1)} }[]<op>{ fun = \LambdaX[mm]{#1[#2]} }
[Supposons une fonction $f\fun<eval>\{t\}$, et \etudions le probl\eme
  o\’u la fonctionnelle $\fun<op>\{S\}\{f\}$ est donn\’ee par\dots]{}
\CcoolClear
\CcoolOption
```

Supposons une fonction $f(t)$, et étudions le problème où la fonctionnelle $S[f]$ est donnée par...

Listing 12. 11 read from file

```
\CcoolRead {\fun<eval>\{t\}}, {\fun<op>\{S\}\{f\}}
\CcoolClear
```

(t), S[f]

Listing 13. CUSUM statistic[]

```

\begin{group}
\CoolOption[ Write = \BooleanTrue ]
\newtheorem{definition}{Definition}
\AfterEndEnvironment{definition}{\CoolHook}
\Cool{
  SuchThat = { ;~ },
  Time = { t },
  Process = { \xi },
  StopT = { T },
  EvalAt = \LambdaX{(#1)}
}
[The CUSUM statistic process and the corresponding one-sided CUSUM
stopping time are defined as follows:
\begin{definition}\label{the CUSUM statistic}. Let~
{
  Scale = { \lambda },
  Real = {\mathcal{R}}
}+*s{{~\in~}}
[~and~]
{ CUSUMthresh = { \nu } }+*c{ $ #1 \in \Real^{+} }$.
[~Define the following processes:]
{
  LogWald = { u },
  CUSUMst = { \StopT_{c} },
  CUSUM = { y },
  LogWaldInf = { m }
}+
[\begin{enumerate}
\item{
  $\LogWald_{\Time}\EvalAt{ \Scale } = \Scale\Process_{\Time} -
\frac{1}{2}\Scale^2\Time$;
  $\LogWaldInf_{\Time}\EvalAt{ \Scale } = \inf_{ 0 \le s \le \Time
}\CUSUM_{s} \EvalAt{ \Scale }$.
}
\item{
  $\CUSUM_{\Time}\EvalAt{ \Scale } = \LogWaldInf_{\Time}\EvalAt{
\Scale } - \LogWald_{\Time}\EvalAt{ \Scale } \ge 0$,
which is the CUSUM statistic process.
}
\item{
  $\CUSUMst \EvalAt{ \Scale, \LogWaldInf } = \inf\left[ \Time \ge 0
\SuchThat \CUSUM_{\Time}\EvalAt{\Scale} \ge \LogWaldInf \right]$,
which is the CUSUM stopping time.
}
\end{enumerate}
\end{definition}\par]{}

```

```
(Check: $\Scale$, $\CUSUM$)
\endgroup
\CoolOption
%
```

The CUSUM statistic process and the corresponding one-sided CUSUM stopping time are defined as follows:

Definition 1 . Let $\lambda \in \mathcal{R}$ and $\nu \in \mathcal{R}^+$. Define the following processes:

1. $u_t(\lambda) = \lambda\xi_t - \frac{1}{2}\lambda^2t$; $m_t(\lambda) = \inf_{0 \leq s \leq t} y_s(\lambda)$.
2. $y_t(\lambda) = m_t(\lambda) - u_t(\lambda) \geq 0$, which is the CUSUM statistic process.
3. $T_c(\lambda, m) = \inf [t \geq 0; y_t(\lambda) \geq m]$, which is the CUSUM stopping time.

(Check: λ, y)

Listing 14. Listing 13 read from file

```
\begingroup
\CoolRead $\Time$ $\Process$ $\Scale$ $\Real$ $\CUSUMthresh$ $\LogWald$
$\CUSUMst$ $\CUSUM$ $\LogWaldInf$
\endgroup
```

$t \xi \lambda \mathcal{R} \nu u T_c y m$

Part IV

Implementation

1 Opening

```
1 \package
2 \c@=ccool
3 \ExplSyntaxOn
```

2 aux

```
\_ccool_aux_inner_set:n #1: \code
4 \cs_new_protected:Nn \_ccool_aux_inner_set:n
5 {
6 \cs_gset:Npn \_ccool_aux_inner:n ##1 {#1}
7 \cs_generate_variant:Nn \_ccool_aux_inner:n { e }
8 }
```

(End definition for _ccool_aux_imer_set:n)

```
\_ccool_aux_key:w #1: \key
```

```

#2 : < value >
 9 \cs_new_protected:Npn \__ccool_aux_key:w #1 = #2 \q_stop
10 {
11   \seq_gput_right:Nx \g__ccool_aux_key_seq { \tl_trim_spaces:n{#1} }
12 }

```

(End definition for __ccool_aux_key:w.)

```

\__ccool_aux_key:n #1 : < key = value >
13 \cs_new_protected:Nn \__ccool_aux_key:n
14 {
15   \__ccool_aux_key:w #1 \q_stop
16 }

```

(End definition for __ccool_aux_key:n.)

```

\__ccool_aux_key:N #1 : < seq >
17 \cs_new_protected:Nn \__ccool_aux_key:N
18 {
19   \seq_gclear_new:N \g__ccool_aux_key_seq
20   \seq_map_function:NN #1 \__ccool_aux_key:n
21 }

```

(End definition for __ccool_aux_key:N.)

```

\__ccool_aux_outer_set:n #1 : < inline code >
22 \cs_new_protected:Nn \__ccool_aux_outer_set:n
23 {
24   \cs_gset:Npn \__ccool_aux_outer:n ##1 {#1}
25 }

```

(End definition for __ccool_aux_outer_set:n.)

```

\__ccool_aux_prop:nn
26 \prop_new:N \g__ccool_aux_prop
27 \cs_new_protected:Nn \__ccool_aux_prop:nn
28 {
29   \prop_gput:Nnn \g__ccool_aux_prop{#1}{#2}
30 }
31 \cs_generate_variant:Nn \__ccool_aux_prop:nn { eo, ee, ex, xo, xe, xx }

```

(End definition for __ccool_aux_prop:nn.)

```

\__ccool_aux_prop:w #1 : < key >
#2 : < value >
32 \tl_new:N \g__ccool_option_expans_tl
33 \cs_new_protected:Npn \__ccool_aux_prop:w #1 = #2 \q_stop
34 {
35   \exp_args:Nx
36   \use:c{__ccool_aux_prop:\g__ccool_option_expans_tl}
37   { \tl_trim_spaces:n{#1} }
38   { \__ccool_aux_inner:n{ \tl_trim_spaces:n{#2} } }
39 }

```

(End definition for `_ccool_aux_prop:w`.)

```
\_ccool_aux_prop:n #1 : < key = value >
40 \cs_new_protected:Nn \_ccool_aux_prop:n
41 {
42   \_ccool_aux_prop:w #1 \q_stop
43 }
```

(End definition for `_ccool_aux_prop:n`.)

```
\_ccool_aux_prop:N #1 : < keyval list >
44 \cs_new_protected:Nn \_ccool_aux_prop:N
45 {
46   \prop_gclear_new:N \g__ccool_aux_prop
47   \seq_if_empty:NTF #1
48   { \c_empty_tl }
49   {
50     \seq_map_function:NN #1 \_ccool_aux_prop:n
51   }
52 }
```

(End definition for `_ccool_aux_prop:N`.)

```
\_ccool_aux_val:Nn #1 : < seq >
#2 : < tl var name >
53 \cs_new_protected:Nn \_ccool_aux_val:Nn
54 {
55   \seq_gclear_new:N \g__ccool_aux_val_seq
56   \_ccool_seq_from_prop:NNn \g__ccool_aux_val_seq #1 { \_ccool_prop_name:n{#2} }
57 }
```

(End definition for `_ccool_aux_val:Nn`.)

```
58 \cs_new:Nn\_ccool_aux_merge:nn{#1#2}
```

3 lang

```
59 \prop_new:N \g__ccool_lang_and_prop
```

`_ccool_lang_and_update:n`

```
60 % \changes{v3.2}
61 % {2021/09/20}
62 % {Replace~\cs[no-index]{erw_prop_keyval:Nn}~by~\cs[no-index]{prop_set_from_keyval:Nn}}
63 \cs_new_protected:Nn \_ccool_lang_and_update:n
64 {
65   \prop_set_from_keyval:Nn
66   \g__ccool_lang_and_prop
67   { #1 }
68 }
69 \cs_generate_variant:Nn \_ccool_lang_and_update:n { e }
```

(End definition for `_ccool_lang_and_update:n`.)

```

\__ccool_lang_and:n
\__ccool_lang_and:
70 \cs_new:Nn \__ccool_lang_and:n
71 {
72   \prop_if_in:NnTF
73   \g__ccool_lang_and_prop
74   {#1}
75   {\prop_item:Nn\g__ccool_lang_and_prop{#1}}
76   {
77     \msg_warning:nnn{__ccool}{lang_and}{#1}
78     \__ccool_lang_and:n{english}
79   }
80 }
81 \ifcsdef{language}
82 {
83   \cs_new:Nn \__ccool_lang_and:{\exp_args:No\__ccool_lang_and:n{language}}
84 }
85 {
86   \cs_new:Nn \__ccool_lang_and:{english}
87 }

```

(End definition for `__ccool_lang_and:n` and `__ccool_lang_and:.`)

```

\c__ccool_lang_and_tl (Note1)
88 \tl_const:Nn \c__ccool_lang_and_tl
89 {
90   %1 https://www.overleaf.com/learn/latex/International_language_support
91   afrikaans=en,
92   basque=eta,
93   catalan=i,
94   croatian=i,
95   czech=a,
96   danish=og,
97   dutch=en,
98   english=and,
99   esperanto=kaj,
100  estonian=ja,
101  finnish=ja,
102  french=et,
103  galician=e,
104  german=und,
105  hungarian='es,
106  icelandic=og,
107  indonesian=dan,
108  irish=agus,
109  italian=e,
110  kurmanji=\^u,
111  latin=et,
112  latvian=un,
113  lithuanian=ir,
114  ngerman=und,
115  polish=i,
116  portuguese=e,

```

¹[todo]: Non latin-alphabet languages

```

117   romanian=\c{s}i,
118   slovak=a,
119   spanish=y,
120   swedish=och,
121   swissgerman=und,
122   turkish=ve,
123   turkmen=we,
124   welsh=a
125 }

```

(End definition for \c_cool_lang_and_tl.)

4 log

_ccool_log_close:

```

126 \iow_new:N \g_ccool_log_iow
127 \AtEndDocument{\iow_close:N \g_ccool_log_iow}
128 \bool_set_false:N \g_ccool_log_open_bool
129 \cs_new_protected:Nn \_ccool_log_close:
130 {
131   \iow_close:N \g_ccool_log_iow
132   \bool_gset_false:N \g_ccool_log_open_bool
133 }

```

(End definition for _ccool_log_close:.)

_ccool_log_open:

```

134 \tl_new:N \g_ccool_log_file_tl
135 \cs_new_protected:Nn \_ccool_log_open:
136 {
137   \tl_gset:Nx \g_ccool_log_to_tl{\g_ccool_log_file_tl}
138   \iow_open:Nn \g_ccool_log_iow {\g_ccool_log_to_tl}
139   \bool_gset_true:N \g_ccool_log_open_bool
140 }

```

(End definition for _ccool_log_open:.)

_ccool_log_read:n #1 : $\langle path \rangle$

```

141 \cs_new_protected:Nn \_ccool_log_read:n
142 {
143   \file_input:n{#1}
144   \tl_log:n{read~from~#1}
145 }
146 \cs_generate_variant:Nn \_ccool_log_read:n { e }

```

(End definition for _ccool_log_read:n.)

_ccool_log_read:

```

147 \cs_new_protected:Nn \_ccool_log_read:
148 {
149   \_ccool_log_read:e{\g_ccool_log_to_tl}
150 }

```

(End definition for _ccool_log_read:.)

`__ccool_log_write:n`

```
151 \tl_new:N \g__ccool_log_to_tl
152 \cs_new_protected:Nn \__ccool_log_write:n
153 {
154   \bool_if:nTF{ \g__ccool_log_open_bool }
155   {
156     \iow_now:Nn \g__ccool_log_iow {#1}
157     \tl_log:n{ write-to~#1 }
158   }
159   { \msg_error:nnn{ __ccool }{ iow }{ \g__ccool_log_iow } }
160 }
161 \cs_generate_variant:Nn \__ccool_log_write:n { e }
```

(End definition for __ccool_log_write:n.)

5 make_key

`__ccool_make_key:Nn`

```
#1 : < token >
#2 : < key >
162 \cs_new_protected:Nn \__ccool_make_key:Nn
163 {
164   \exp_args:NNx
165   \DeclareDocumentCommand{#1}
166   { D<>{\g__ccool_option_param_tl} }
167   {
168     \__ccool_prop_item:nn{##1}{#2}
169   }
170 }
171 \cs_generate_variant:Nn \__ccool_make_key:Nn {c}
```

(End definition for __ccool_make_key:Nn.)

`__ccool_make_key:n`

```
#1 : < key >
172 \cs_new_protected:Nn \__ccool_make_key:n
173 {
174   \__ccool_make_key:cn{#1}{#1}
175 }
176 \cs_generate_variant:Nn \__ccool_make_key:n { e }
```

(End definition for __ccool_make_key:n.)

`__ccool_make_key:N`

```
#1 : < seq >
177 \cs_new_protected:Nn \__ccool_make_key:N
178 {
179   \seq_map_function:NN #1 \__ccool_make_key:e
180 }
```

(End definition for __ccool_make_key:N.)

6 make_ccool

_ccool_make_ccool_exp:nnn

```

181 %      ^^A      \erw_seq_use:Nn
182 \cs_new_protected:Nn \_ccool_make_ccool_exp:nnn
183 {
184   \_ccool_aux_val:Nn \g__ccool_aux_key_seq {#1}
185   \_ccool_aux_outer_set:n{#3}
186   \_ccool_aux_outer:n
187   {
188     \exp_args:NNf
189     \_ccool_seq_use:Nn
190     \g__ccool_aux_val_seq
191     {#2}
192   }
193 }

```

(End definition for _ccool_make_ccool_exp:nnn.)

_ccool_make_ccool_key:nnn

```

194 \cs_new_protected:Nn \_ccool_make_ccool_key:nnn
195 {
196   \_ccool_prop_if_exist:nTF{#1}
197   { \c_empty_tl }
198   { \_ccool_prop_new:n{#1} }
199   \exp_args:No \_ccool_aux_inner_set:n{#2}
200   \seq_set_from_clist:Nn \g__ccool_aux_keyval_seq {#3}
201   \_ccool_aux_prop:N \g__ccool_aux_keyval_seq
202   \_ccool_prop_append:Nn \g__ccool_aux_prop {#1}
203   \_ccool_aux_key:N \g__ccool_aux_keyval_seq
204   \_ccool_make_key:N \g__ccool_aux_key_seq
205 }

```

(End definition for _ccool_make_ccool_key:nnn.)

_ccool_make_ccool_sideeffect:nnn

```

[5]
206 \cs_new_protected:Nn \_ccool_make_ccool_sideeffect:nnn
207 {
208   \_ccool_make_ccool_key:nnn{#1}{#2}{#3}
209   \bool_if:nTF{ \g__ccool_log_open_bool }
210   {
211     \_ccool_log_write:n
212     {
213       \begingroup
214       \def \_ccool_log_entry { \Ccool<#1>c{#2}{#3} } \expandafter
215       \endgroup \_ccool_log_entry
216     }
217   }{\c_empty_tl}
218 }

```

(End definition for _ccool_make_ccool_sideeffect:nnn.)

```

\_ccool_make_ccool:nnnn #1 : < token list >
                        #2 : < seq1 >
                        #3 : < seq2 >

```

```

#4 : < prop >
219 \cs_new_protected:Npn \__ccool_make_ccool:nnnn #1 #2 #3 #4
220 {
221   \exp_args:NNx \DeclareDocumentCommand \Ccool
222     {%^~A      2      3      4 5 6      7 8 9
223     +o D<>{#1} E{ c }{#{2}} m t+ s E{ s c }{#{3}{#4}} +o
224     }
225     {
226     \IfValueT{##1}{##1}
227     \__ccool_make_ccool_sideeffect:nnn{##2}{##3}{##4}
228     \IfBooleanT{##6}
229     {
230     \__ccool_make_ccool_exp:nnn{##2}{##7}{##8}
231     }
232     \bool_if:nTF{##5}
233     {
234     \gappto{\CcoolHook}
235     {
236     \__ccool_make_ccool_sideeffect:nnn{##2}{##3}{##4}
237     }
238     }
239     {\c_empty_tl}
240     \IfValueT{##9}
241     {
242     \exp_not:n{ \Ccool[##9] }
243     }
244     }
245 }

```

(End definition for __ccool_make_ccool:nnnn.)

7 msg

```

246 \msg_new:nnn {__ccool}
247 { iow }
248 {#1~is~closed~can't~write}
249 \msg_new:nnn {__ccool}
250 {lang_and}
251 {-key~#1~missing~for~global~option~'And';~falling~back~on~'english'}

```

8 option

```

\__ccool_option_inner:n #1 : <code>
252 \tl_new:N \g__ccool_option_inner_tl
253 \cs_new_protected:Nn \__ccool_option_inner:n
254 {
255   \tl_gset:Nn \g__ccool_option_inner_tl {#1}
256 }

```

(End definition for __ccool_option_inner:n.)

```

\__ccool_option_param:n #1 : <token list>
257 \tl_new:N \g__ccool_option_param_tl

```

```

258 \cs_new_protected:Nn \__ccool_option_param:n
259 {
260   \tl_gset:Nn \g__ccool_option_param_tl{#1}
261 }

```

(End definition for __ccool_option_param:n.)

__ccool_option_outer:n #1 : *< inline code >*

```

262 \tl_new:N \g__ccool_option_outer_tl
263 \cs_new_protected:Nn \__ccool_option_outer:n
264 {
265   \tl_gset:Nn \g__ccool_option_outer_tl {#1}
266 }

```

(End definition for __ccool_option_outer:n.)

__ccool_option_separ:n #1 : *<{ tl₁ }><{ tl₂ }><{ tl₃ }>*

```

267 \tl_new:N \g__ccool_option_separ_tl
268 \cs_new_protected:Nn \__ccool_option_separ:n
269 {
270   \cs_gset:Npn \g__ccool_option_separ_tl {#1}
271 }

```

(End definition for __ccool_option_separ:n.)

\g__ccool_option_separ_tl

```

272 \ifcsdef{text}
273 {
274   \tl_const:Nn \c__ccool_option_separ_default_tl
275   {
276     { \text{\ } \__ccool_lang_and:{\ } }
277     { \text{,{\ } } }
278     { \text{,{\ } \__ccool_lang_and:{\ } } }
279   }
280 }
281 {
282   \tl_const:Nn \c__ccool_option_separ_default_tl
283   {
284     { {\ } \__ccool_lang_and:{\ } }
285     { ,{\ } }
286     { ,{\ } \__ccool_lang_and:{\ } }
287   }
288 }

```

(End definition for \g__ccool_option_separ_tl.)

9 prop

__ccool_prop_append:NN #1 : *< prop₁ >*

```

#2 : < prop2 >
289 \cs_new_protected:Npn \__ccool_prop_append:NN #1 #2
290 {
291   \cs_set:Nn \__ccool_prop_append:nn
292   {
293     \prop_gput:Nnx #1 {##1}{ \prop_item:Nn #2{##1} }
294   }
295   \prop_map_function:NN #2 \__ccool_prop_append:nn
296 }
297 \cs_generate_variant:Nn \__ccool_prop_append:NN { cN }

```

(End definition for __ccool_prop_append:NN.)

```

\__ccool_prop_append:Nn #1 : < prop >
#2 : < tl var name >
298 \cs_new_protected:Nn \__ccool_prop_append:Nn
299 {
300   \__ccool_prop_append:cN{ \__ccool_prop_name:n {#2} } #1
301 }

```

(End definition for __ccool_prop_append:Nn.)

```

\__ccool_prop_clear_new:n #1 : < tl var name >
302 \cs_new_protected:Nn \__ccool_prop_clear_new:n
303 {
304   \exp_args:No \prop_clear_new:c{ \__ccool_prop_name:n {#1} }
305 }

```

(End definition for __ccool_prop_clear_new:n.)

```

\__ccool_prop_clear_new_map:n #1 : < keyval list >
306 \cs_new_protected:Nn \__ccool_prop_clear_new_map:n
307 {
308   \seq_set_from_clist:Nn \g__ccool_aux_key_seq {#1}
309   \seq_map_function:NN \g__ccool_aux_key_seq \__ccool_prop_clear_new:n
310 }

```

(End definition for __ccool_prop_clear_new_map:n.)

```

\__ccool_prop_if_exist:nTF #1 : < tl1 >
#2 : < tl2 >
#3 : < tl3 >
311 \cs_new:Nn \__ccool_prop_if_exist:nTF
312 {
313   \prop_if_exist:cTF{ \__ccool_prop_name:n {#1} }{#2}{#3}
314 }

```

(End definition for __ccool_prop_if_exist:nTF.)

```

\__ccool_prop_item:nn #1 : < tl var name >
#2 : < key >
315 \cs_new:Nn \__ccool_prop_item:nn
316 {
317   \prop_item:cn { \__ccool_prop_name:n {#1} } {#2}
318 }

```

(End definition for `_ccool_prop_item:n`.)

```
\_ccool_prop_name:n #1 : < tl var name >
319 \cs_new:Npn \_ccool_prop_name:n #1{ \_ccool_#1 }
(End definition for \_ccool_prop_name:n.)
```

```
\_ccool_prop_new:n #1 : < tl var name >
320 \cs_new_protected:Nn \_ccool_prop_new:n
321 {
322   \prop_new:c{ \_ccool_prop_name:n {#1} }
323 }
(End definition for \_ccool_prop_new:n.)
```

10 seq

```
\_ccool_seq_from_prop:NNn #1 : < seq1 >
#2 : < seq2 > (keys)
#3 : < prop >
324 \cs_new_protected:Nn \_ccool_seq_from_prop:NNn
325 {
326   \cs_set_protected:Nn \_ccool_seq_from_prop:n
327   {
328     \seq_gput_right:No #1 { \prop_item:cn{#3}{##1} }
329   }
330   \seq_map_function:NN #2 \_ccool_seq_from_prop:n
331 }
(End definition for \_ccool_seq_from_prop:NNn.)
```

11 seq_use

```
\_ccool_seq_from_prop:NNn
332 % \changes{v3.2}
333 % {2021/09/20}
334 % {Added~\cs[no-index]{\_ccool_seq_use:Nn}~in~replacement~of~\cs[no-index][erw_seq_use:Nn]}
335 \msg_new:nnn{\_ccool}{separ}{#1~expects~1~to~3~items,~#2}
336 \cs_new:Nn \_ccool_seq_use:Nn
337 {
338   \exp_last_unbraced:NNf
339   \seq_use:Nnnn #1
340   \_ccool_tl_separators:n{#2}
341 }
342 \cs_new:Nn \_ccool_tl_separators:n
343 { \_ccool_tl_separators:en{ \tl_count:n{#1} }{#1} }
344 \cs_new:Nn \_ccool_tl_separators:nn
345 { \int_case:nnTF {#1}
346   { {1}
347     { \prg_replicate:nn{ 3 }{#2} }
348     {2}
349     {
```

```

350     { \use_ii:nn #2 }
351     { \use_i:nn #2 }
352     { \use_i:nn #2 \use_ii:nn #2 }
353   }
354   {3}{#2}
355 }
356 { \c_empty_tl }
357 {
358   \msg_error:nnnn { __ccool }
359   { separ }
360   { \__ccool_tl_separators:nn }
361   {#2}
362 }
363 }
364 \cs_generate_variant:Nn \__ccool_tl_separators:nn { e }

```

(End definition for __ccool_seq_from_prop:NNn.)

12 Front-end

\CcoolClearimpl:cs:clear

\CcoolClear

```

365 \NewDocumentCommand{ \CcoolClear }
366 { D<>{\g__ccool_option_param_tl} }
367 {
368   \__ccool_prop_clear_new_map:n{#1}
369 }

```

(End definition for \CcoolClear. This function is documented on page 4.)

\CcoolHookimpl:cs:hook

\CcoolHook

```

370 \NewDocumentCommand{\CcoolHook}{-}{\c_empty_tl}

```

(End definition for \CcoolHook. This function is documented on page 4.)

\CcoolLambdaimpl:cs:lambda

\CcoolLambda (*Note*²)

```

371 % \changes{v3.2}
372 % {2021/09/20}{\cs[CcoolLambda]'s-implementation-switched-from-\pkg{erw-l3}-to-\pkg{lamt}
373 \ProvideDocumentCommand \CcoolLambda { 0{m} m }
374 { \lambdax:nn{#1}{#2} }

```

(End definition for \CcoolLambda. This function is documented on page ??.)

\CcoolOptionimpl:cs:option

\CcoolOption (*Note*³) (*Note*⁴)

```

375 \NewDocumentCommand{ \CcoolOption }
376 { 0{ And, Expans, File, Inner, Param, Outer, Separ, Write } }
377 {

```

²[todo]: allow only m- or o-type arguments

³[todo]: Fix placeholders passed to options requiring code (only one pound sign)

⁴[abandon]: Requirement: write to file if Write; Update: redundant with \cs {Ccool}+Write

```

378 \keys_set:nn{ __ccool }{#1}
379 }

(End definition for \CcoolOption. This function is documented on page 4.)

380 \keys_define:nn { __ccool }
381 {
382 And .code:n = { \__ccool_lang_and_update:e{ #1 } },
383 And .default:n = { \c__ccool_lang_and_tl },
384 And .initial:n = { \c__ccool_lang_and_tl },
385 Expans .multichoices:nn = { eo, ee, ex, xo, xe, xx }
386 { \tl_gset_eq:NN \g__ccool_option_expans_tl \l_keys_choice_tl },
387 Expans .default:n = { xo },
388 Expans .initial:n = { xo },
389 % \changes{v3.2}
390 % {2021/09/20}
391 % {Removed-module~key~File's~relianced-on~a~timestamp~(clumsy)}
392 File .code:n = { \tl_gset:Nx \g__ccool_log_file_tl{#1} } }
393 \cs_new_protected:Nn
394 \__ccool_keys_define_file:n
395 { \keys_define:nn { __ccool }
396   { File .code:n = { \tl_gset:Nx \g__ccool_log_file_tl{#1} },
397     File .default:n = { #1 },
398     File .initial:n = { #1 } } }
399 \cs_generate_variant:Nn\__ccool_keys_define_file:n{e}
400 \__ccool_keys_define_file:e
401 { \exp_args:Ne\__ccool_aux_merge:nn{\c_sys_jobname_str}{__ccool_log}}
402 \keys_define:nn { __ccool }
403 {
404   Inner .code:n={
405     \__ccool_option_inner:n{#1}
406     \exp_last_unbraced:Nf
407     \__ccool_make_ccool:nnnn
408     {
409       { \g__ccool_option_param_tl }
410       { \g__ccool_option_inner_tl }
411       { \g__ccool_option_separ_tl }
412       { \g__ccool_option_outer_tl }
413     }
414   },
415   Inner .value_required:n = false,
416   Inner .default:n = {###1},
417   Inner .initial:n = {###1},
418   Param .code:n={
419     \__ccool_option_param:n{#1}
420     \exp_last_unbraced:Nf
421     \__ccool_make_ccool:nnnn
422     {
423       { \g__ccool_option_param_tl }
424       { \g__ccool_option_inner_tl }
425       { \g__ccool_option_separ_tl }
426       { \g__ccool_option_outer_tl }
427     }

```

```

428 },
429 Param .value_required:n = false,
430 Param .default:n = { Default },
431 Param .initial:n = { Default },

432 Outer .code:n={
433   \_ccool_option_outer:n{#1}
434   \exp_last_unbraced:Nf
435   \_ccool_make_ccool:nmnn
436   {
437     { \g_ccool_option_param_tl }
438     { \g_ccool_option_inner_tl }
439     { \g_ccool_option_separ_tl }
440     { \g_ccool_option_outer_tl }
441   }
442 },
443 Outer .value_required:n = false,
444 Outer .default:n = { \ensuremath{####1} },
445 Outer .initial:n = { \ensuremath{####1} },

446 Separ .code:n={
447   \_ccool_option_separ:n{#1}
448   \exp_last_unbraced:Nf
449   \_ccool_make_ccool:nmnn
450   {
451     { \g_ccool_option_param_tl }
452     { \g_ccool_option_inner_tl }
453     { \g_ccool_option_separ_tl }
454     { \g_ccool_option_outer_tl }
455   }
456 },
457 Separ .value_required:n = false,
458 Separ .default:n = { \c_ccool_option_separ_default_tl },
459 Separ .initial:n = { \c_ccool_option_separ_default_tl },

460 Write .code:n = {
461   \bool_if:nTF{#1}
462   {\_ccool_log_open:}
463   {\_ccool_log_close:}
464 },
465 Write .value_required:n = false,
466 Write .default:n = \BooleanFalse,
467 Write .initial:n = \BooleanFalse

468 }

```

\CcoolRead

```

469 \NewDocumentCommand{\CcoolRead}
470 {o}
471 {
472   \IfValueTF{#1}
473   {\_ccool_log_read:e{#1}}
474   {\_ccool_log_read:}
475 }

```

(End definition for \CcoolRead. This function is documented on page 5.)

\CcoolVers

```
476 \NewDocumentCommand{\CcoolVers}  
477 {}  
478 {\use:c{ver@ccool.sty}}
```

(End definition for \CcoolVers. This function is documented on page 5.)

13 Closing

```
479 \ExplSyntaxOff  
480 </package>
```

Change History

v1.0	General: Initial version	6	Rename: <code>\OpsOption</code> to <code>\CcoolOption</code>	6
v1.1	General: Add: <code>Save</code>	6	Rename: <code>\OpsRead</code> to <code>\CcoolRead</code>	6
	Add: <code>\OpsRestore</code>	6	Rename: <code>\Ops</code> to <code>\Ccool</code>	6
	Add: <code>\OpsTest</code>	6	Rename: <code>oops</code> to <code>ccool</code> (better describes the purpose)	6
	Rearrange: much of the implementation	6	v1.7	General: Delete: <code>\CcoolDebug</code>
	Replace: <code>\OpsOptions</code> by <code>\OpsOption</code>	6	v1.8	General: Add: <code>\CcoolVers</code>
	Replace: <code>{<keyval>}</code> by <code><keyval></code> given that option type <code>G</code> not recommended ^[4]	6		Add: <code>\CcoolLambda</code>
	Replace: <code>GenericObject</code> by <code>Name</code>	6	v1.9	General: Add: support for LuaTeX
	Replace: <code>Separators</code> by <code>Separ</code>	6		Move: from Part I to Part IV , what is now that part's section 12
v1.2	General: Add: optional <code>*to \OpsNew</code> as instruction to expand keyval list ₁	6	v2.0	General: Add: support for XeTeX
	Delete: <code>\OpsTest</code>	6		Delete: <code>File</code> 's dependency on <code>texosquery</code> and <code>\pdfcreationdate</code>
	Delete: <code><keyval></code> and <code><code2></code>	6		Update: <code>\RequirePackage</code> , <code>\NeedsTeXFormat</code> 's second argument / TeX Live 2020
	Replace: <code>\OpsClear{<tl2>}</code> by <code>\OpsClear[<keyval list>]</code>	6	v2.1	General: Replace: <code><tl2></code> 's position within <code>\Ccool</code> 's argument list, from first to second. Greater versatility
	Replace: <code>\Restore</code> by <code>\Read</code>	6		Replace: <code>\CcoolLambda</code> 's optional integer argument (number of <code>m</code> 's) by a standard argument list
	Replace: <code>\Save</code> by <code>\Write</code>	6		Replace: <i>global option Name</i> by <code>Param</code>
v1.3	General: Replace: <code>\OpsNew</code> by <code>\Ops</code>	6		Replace: as the default of <code>Param</code> , <code>Math</code> by <code>Default</code>
	Replace: <code>{<tl2>}</code> and <code>[<tl2>]</code> by <code><tl2></code>	6	v2.2	General: Replace: part of the abstract's with more straightforward descriptions based on input from forum participants
v1.4	General: Add: section 2	6	v2.3	General: Rearranged: <code>\Ccool</code> 's subsections. Previously, by argument. Now, by feature.
	Add: <code>\OpsDebug</code>	6		Replace: for <code>\Ccool</code> , <code>i{}</code> by <code>c{}</code>
	Add: <code>\OpsHook</code>	6		Replace: In step 2), the created command's implementation, from <code>\ProvideDocumentCommand</code> to <code>\DeclareDocumentCommand</code>
	Add: <code>Expans</code> (for debugging' sake, but...)	6		v2.5
	Add: optional <code>+to \OpsNew</code> to make side effects persist beyond local group	6		General: Modify: behavior of Part I Expand the <i>val</i>'s , rely on <code>erw-l3's \erw_seq_use:Nn</code>
	Replace: <code>s{<tl3>}{<tl4>}{<tl5>}</code> by <code>s{<tl3>}{<tl3>}{<tl4>}{<tl3>}{<tl4>}{<tl5>}</code>	6		
v1.5	General: Add: <code>File</code>	6		
	Delete: dependence on <code>datetime</code>	6		
v1.6	General: Rename: <code>\OpsClear</code> to <code>\CcoolClear</code>	6		
	Rename: <code>\OpsDebug</code> to <code>\CcoolDebug</code>	6		
	Rename: <code>\OpsHook</code> to <code>\CcoolHook</code>	6		

Modify: Rely on erw-l3's	<code>\usepackage</code>	6	6
<code>\erw_jobnametimestamp</code> :	v3.2	6	
v2.6	General: Modify: Rely on erw-l3's		
<code>\erw_lambda:nmn</code>	<code>\begingroup</code> and <code>\endgroup</code> in-	6	side listings.
v2.7	General: Add: <i>global option</i> <code>And</code>	6	<code>\texorpdfstring</code> was used with one ar-
Modify: <code>Separ</code> 's default rely on	argument instead of two in listing ti-		les, so removed it.
<code>babel</code> and <code>amsmath</code> , if applicable . .	Added dependence on <code>lambdax</code>	6	11
Modify: Replace 'm'-type	Removed dependence on erw-l3.	6	11
argument by 'o'-type argument . . .	Removed listing 'Hello, world (for	6	testing's
v2.8	sake)'. That should be for l3build . .		8
General: Fix: conflict between	Removed listing changes (little infor-		6
<code>\usepackage[spanish]{babel}</code> and	mation, difficult to keep track)	6	
Parameterize the <i>key</i> 's	Removed us-		
v2.9	age for <code>\CcoolLambda</code> as super-		
General: Miscellaneous	seded by <code>\LambdaX</code>	6	4
v3.0	General: Miscellaneous	6	Updated list-
v3.1	General: Replaced: Listing '??'s		ings with <code>\LambdaX</code> in place of <code>\CcoolLambda</code>
content, from exhaustive	dependencies to those explicit with	
			9
			Updated listing 'Fonction et fonc-
			tionelle', with use of parameter
			9

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		bool commands:
<code>\'</code>	105	<code>\bool_gset_false:N</code>	132
<code>\langle key_i <tl_2 ></code>	4	<code>\bool_gset_true:N</code>	139
<code>\langle key_i \rangle</code>	3, 6	<code>\bool_if:nTF</code>	154, 209, 232, 461
<code>\langle key_i \rangle</code>	3	<code>\bool_set_false:N</code>	128
<code>And</code> (option)	4	<code>\BooleanFalse</code>	5, 466, 467
<code>Expans</code> (option)	4	<code>\BooleanTrue</code>	4, 5
<code>File</code> (option)	4		
<code>Inner</code> (option)	5		
<code>Outer</code> (option)	5		
<code>Param</code> (option)	5		
<code>Separ</code> (option)	5		
<code>Write</code> (option)	5		
<code>\^</code>	110		
<code>_</code>	276, 277, 278, 284, 285, 286		
	A		C
<code>\AtEndDocument</code>	127	<code>\c</code>	117
		<code>\Ccool</code>	3, 4, 7, 26, 214, 221, 242
		ccool internal commands:	
		<code>__ccool_aux_inner:n</code>	6, 7, 38
		<code>__ccool_aux_inner_set:n</code>	4, 199
		<code>__ccool_aux_key:N</code>	17, 203
		<code>__ccool_aux_key:n</code>	13, 20
		<code>__ccool_aux_key:w</code>	9, 15
		<code>\g__ccool_aux_key_seq</code>	
		11, 19, 184, 204, 308, 309
		<code>\g__ccool_aux_keyval_seq</code>	200, 201, 203
		<code>__ccool_aux_merge:nm</code>	58, 401
		<code>__ccool_aux_outer:n</code>	24, 186
	B		
<code>\begingroup</code>	213		

__ccool_aux_outer_set:n	22, 185
g__ccool_aux_prop	26, 29, 46, 202
__ccool_aux_prop:N	44, 201
__ccool_aux_prop:n	40, 50
__ccool_aux_prop:nn	26
__ccool_aux_prop:w	32, 42
__ccool_aux_val:Nn	53, 184
g__ccool_aux_val_seq	55, 56, 190
__ccool_keys_define_file:n	394, 399, 400
__ccool_lang_and: 70, 276, 278, 284, 286	
__ccool_lang_and:n	70
g__ccool_lang_and_prop	59, 66, 73, 75
c__ccool_lang_and_tl	88, 383, 384
__ccool_lang_and_update:n	60, 382
__ccool_log_close:	126, 463
__ccool_log_entry	214, 215
g__ccool_log_file_tl	134, 137, 392, 396
g__ccool_log_iow	126, 127, 131, 138, 156, 159
__ccool_log_open:	134, 462
g__ccool_log_open_bool	128, 132, 139, 154, 209
__ccool_log_read:	147, 474
__ccool_log_read:n	141, 149, 473
g__ccool_log_to_tl	137, 138, 149, 151
__ccool_log_write:n	151, 211
__ccool_make_ccool:nnnn	219, 407, 421, 435, 449
__ccool_make_ccool_exp:nnn	181, 230
__ccool_make_ccool_key:nnn	194, 208
__ccool_make_ccool_sideeffect:nnn	206, 227, 236
__ccool_make_key:N	177, 204
__ccool_make_key:n	172, 179
__ccool_make_key:Nn	162, 174
g__ccool_option_expans_tl	32, 36, 386
__ccool_option_inner:n	252, 405
g__ccool_option_inner_tl	252, 255, 410, 424, 438, 452
__ccool_option_outer:n	262, 433
g__ccool_option_outer_tl	262, 265, 412, 426, 440, 454
__ccool_option_param:n	257, 419
g__ccool_option_param_tl	166, 257, 260, 366, 409, 423, 437, 451
__ccool_option_separ:n	267, 447
c__ccool_option_separ_default_tl	274, 282, 458, 459
g__ccool_option_separ_tl	267, 270, 272, 411, 425, 439, 453
__ccool_prop_append:NN	289, 300
__ccool_prop_append:Nn	202, 298
__ccool_prop_append:nn	291, 295
__ccool_prop_clear_new:n	302, 309
__ccool_prop_clear_new_map:n	306, 368
__ccool_prop_if_exist:nTF	196, 311
__ccool_prop_item:nn	168, 315
__ccool_prop_name:n	56, 300, 304, 313, 317, 319, 322
__ccool_prop_new:n	198, 320
__ccool_seq_from_prop:n	326, 330
__ccool_seq_from_prop:NNn	56, 324, 332
__ccool_seq_use:Nn	189, 336
__ccool_tl_separators:n	340, 342
__ccool_tl_separators:nn	343, 344, 360, 364
\CcoolClear	4, 22, 365
\CcoolHook	3, 4, 22, 234, 370
\CcoolLambda	22, 27, 371
\CcoolOption	4, 22, 375
\CcoolRead	5, 6, 469
\CcoolVers	5, 476
\changes	60, 332, 371, 389
\CoolHook	6
\cs	62, 334, 372
cs commands:	
\cs_generate_variant:Nn	7, 31, 69, 146, 161, 171, 176, 297, 364, 399
\cs_gset:Npn	6, 24, 270
\cs_new:Nn	58, 70, 83, 86, 311, 315, 336, 342, 344
\cs_new:Npn	319
\cs_new_protected:Nn	4, 13, 17, 22, 27, 40, 44, 53, 63, 129, 135, 141, 147, 152, 162, 172, 177, 182, 194, 206, 253, 258, 263, 268, 298, 302, 306, 320, 324, 393
\cs_new_protected:Npn	9, 33, 219, 289
\cs_set:Nn	291
\cs_set_protected:Nn	326
D	
\DeclareDocumentCommand	165, 221
\def	214
E	
\endgroup	215
\ensuremath	444, 445
erw commands:	
\erw_seq_use:Nn	181
exp commands:	
\exp_args:Ne	401
\exp_args:NMf	188
\exp_args:NNx	164, 221
\exp_args:No	83, 199, 304

\exp_args:Nx	35	Expans	4
\exp_last_unbraced:Nf	406, 420, 434, 448	File	4
\exp_last_unbraced:NNf	338	Inner	5
\exp_not:n	242	Outer	5
\expandafter	214	Param	5
\ExplSyntaxOff	479	Separ	5
\ExplSyntaxOn	3	Write	5
F		P	
file commands:		\pkg	372
\file_input:n	143	prg commands:	
\function	3	\prg_replicate:nn	347
G		prop commands:	
\gappto	234	\prop_clear_new:N	304
I		\prop_gclear_new:N	46
\IfBooleanT	228	\prop_gput:Nnn	29, 293
\ifcsdef	81, 272	\prop_if_exist:NTF	313
\IfValueT	226, 240	\prop_if_in:NnTF	72
\IfValueTF	472	\prop_item:Nn	75, 293, 317, 328
int commands:		\prop_map_function:NN	295
\int_case:nnTF	345	\prop_new:N	26, 59, 322
iow commands:		\prop_set_from_keyval:Nn	65
\iow_close:N	127, 131	\ProvideDocumentCommand	373
\iow_new:N	126	Q	
\iow_now:Nn	156	quark commands:	
\iow_open:Nn	138	\q_stop	9, 15, 33, 42
K		R	
keys commands:		\Real	3
\l_keys_choice_tl	386	S	
\keys_define:nn	380, 395, 402	seq commands:	
\keys_set:nn	378	\seq_gclear_new:N	19, 55
L		\seq_gput_right:Nn	11, 328
\LambdaX	27	\seq_if_empty:NTF	47
\lambdax	3	\seq_map_function:NN	20, 50, 179, 309, 330
lambdax commands:		\seq_set_from_clist:Nn	200, 308
\lambdax:nn	374	\seq_use:Nnnn	339
\languagename	83	sys commands:	
M		\c_sys_jobname_str	401
msg commands:		T	
\msg_error:nnn	159	\text	276, 277, 278
\msg_error:nnnn	358	tl commands:	
\msg_new:nnn	246, 249, 335	\c_empty_tl	48, 197, 217, 239, 356, 370
\msg_warning:nnn	77	\tl_const:Nn	88, 274, 282
N		\tl_count:n	343
\NewDocumentCommand	7, 365, 370, 375, 469, 476	\tl_gset:Nn	137, 255, 260, 265, 392, 396
O		\tl_gset_eq:NN	386
options:		\tl_log:n	144, 157
And	4	\tl_new:N	32, 134, 151, 252, 257, 262, 267
		\tl_trim_spaces:n	11, 37, 38

	U			
use commands:		<code>\use_i:nn</code>	351, 352	
	<code>\use:N</code>	36, 478	<code>\use_ii:nn</code>	350, 352