

The ccool package*

Erwann Rogard†

Released 2020/10/10

Abstract

The package `ccool` for L^AT_EX is a *key-value* interface, `\Ccool`, on top of `xparse`'s document command parser. Global options control input processing and its expansion. By default, they are set to meet likely requirements, depending on context: the selected language, and which of text and math mode is active. These options can be overridden inline. This versatility could find its use, for example, to encode notational conventions (such as `\Real` \rightarrow `\mathbb{R}`) at the point where they are introduced in the `document` (“Let \mathbb{R} denote real numbers”). Polymorphic commands can be generated by parameterizing the keys (for instance, one parameter value for style, another for a property). User input to `\Ccool` can optionally be serialized. This can be useful for typesetting documents sharing the same notation.

Résumé

L'extension `ccool` pour L^AT_EX met à disposition une interface de type *clé-valeur*, `\Ccool`, destinée à faciliter la génération de commandes. Les paramètres globaux contrôlant le traitement de ces *clé-valeur* sont fixés par défaut pour répondre aux besoins courants, suivant le contexte (langage, mode textuel ou mathématique). Un exemple d'application, est la command-isation des conventions de notation (`\Reel` \rightarrow `\mathbb{R}`), au point dans le `document` où elles sont introduites (“Soit \mathbb{R} les nombres réels.”). Des commandes polymorphes peuvent être générées, en associant aux clés un paramètre (par exemple, une valeur pour le style typographique, une autre pour la description du concept associé). En option, les instructions passées à cette interface peuvent être sauvegardées, ce qui peut être utile pour la rédaction de documents faisant appel à des conventions typographiques communes.

Contents

I	Usage	4
0	Convention	4
1	Loading the package	5

*This file describes version v3.1, last revised 2020/10/10.

†firstname dot lastname AusTria gmail dot com

2	\Ccool	5
2.1	Core feature	5
2.2	Process the <i>val_is</i>	5
2.3	Append to a hook	5
2.4	Expand the <i>val_is</i>	6
2.5	Head	6
2.6	Tail	6
2.7	Parameterize the <i>key_is</i>	6
2.8	Write	6
3	\CcoolClear	6
4	\CcoolHook	6
5	\CcoolLambda	6
6	\CcoolOption	7
6.1	And	7
6.2	Expans	7
6.3	File	7
6.4	Inner	7
6.5	Param	7
6.6	Outer	7
6.7	Separ	7
6.8	Write	7
7	\CcoolRead	8
8	\CcoolVers	8
9	Do's and dont's	8
II	Listing	10
1.	\CcoolVers	10
2.	This doc's dependencies	10
3.	Let \mathbb{N} and \mathbb{R} denote. . . start of the tutorial)	10
4.	Equivalent to 3, with \NewDocumentCommand	10
5.	Equivalent to 4, with \Ccool	10
6.	Equivalent to 5, with expansion	11
7.	Equivalent to 5, parameterized (end of the tutorial)	11
8.	Language and mode	11
9.	Separators	12

10. Hello, world! (for testing's sake)	12
11. Listing 10 read from file	13
12. Probability space	13
13. Listing 12 read from file	13
14. Mittelwertsatz für n variable.	14
15. Listing 14 read from file	14
16. Families of polynomial functions	14
17. Listing 16 read from file	15
18. Same as Listing 16, but arbitrary number system	15
17. Listing 18 read from file	15
20. Fonction et fonctionelle	16
21. Listing 20 read from file	16
22. CUSUM statistic.	16
23. Listing 22 read from file	17
III Other	19
1 Acknowledgment	19
2 Genealogy	19
3 Install	19
4 Issue	19
5 Support	19
6 Testing	19
6.1 Technicality	19
6.2 Platform	20
6.3 Engine	20
6.4 Results	20
6.5 Other	20
7 To do	20
8 References	20

Change History	21
Index	23
IV Implementation	26
1 Opening	26
2 aux	26
3 lang	28
4 log	29
5 make_key	30
6 make_ccool	31
7 msg	32
8 option	33
9 prop	34
10 seq	35
11 Front-end	36
11.1 \CcoolClear	36
11.2 \CcoolHook	36
11.3 \CcoolLambda	36
11.4 \CcoolOption	36
11.4.1 And	36
11.4.2 Expans	36
11.4.3 File	37
11.4.4 Inner	37
11.4.5 Param	37
11.4.6 Outer	37
11.4.7 Separ	38
11.4.8 Write	38
11.5 \CcoolRead	38
11.6 \CcoolVers	38

Part I Usage

Convention

- a) Loosely, those of [2], for example as to the meaning of *token list*, which may be abbreviated *tl*.
- b) Those of [5], for example [arg] is an ‘*o*’-type argument.
- c) $\langle X \rangle \leftarrow Y$: set $\langle X \rangle$ to Y
- d) $\backslash X \rightarrow Y$: $\backslash X$ expands to Y
- e) If unspecified, the environment in which a macro is to be used is **document**.

```
\usepackage \usepackage{ccool}
```

Requirement

1. `ccool.sty` is in the path of the L^AT_EX engine. See [Part III, section 5](#).
2. Put in the *preamble*

```
\Ccool \Ccool[ $\langle tl_1 \rangle$ ] $\langle tl_2 \rangle$ >c{ $\langle code_1 \rangle$ }{ $\langle keyval list_1 \rangle$ }+*s{ $\langle separators \rangle$ }c{ $\langle code_2 \rangle$ }[ $\langle tl_6 \rangle$ ]
```

where $\langle separators \rangle$ is either of: $\{\langle tl_3 \rangle\}$, $\{\langle tl_3 \rangle\}\{\langle tl_4 \rangle\}$, and $\{\langle tl_3 \rangle\}\{\langle tl_4 \rangle\}\{\langle tl_5 \rangle\}$.

Semantics See [subsection 2.1-2.8](#).

2.1 Core feature

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

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

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

See `\CcoolLambda` to allow command $\backslash \langle key_i \rangle$ to take arguments.

2.2 Process the val_i s

`\Ccool c{<code1>}{<keyval list1>}` 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”.

2.3 Append to a hook

`\Ccool{<keyval list1>+}` 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**).

2.4 Expand the val_i s

`\Ccool{<keyval list1>*}` supplements the **Core feature** with the expansion of the $\langle val_i \rangle$'s using typesetting rules encoded in *global option* **Separ** and **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{<separators>}` and `c{<code2>}`, respectively, to the list of arguments.

2.5 Head

`\Ccool[<tl1>]{<keyval list1>}` expands $\langle tl_1 \rangle$ and executes the **Core feature**.

There may be situations where it is convenient to pass $\langle tl_1 \rangle$ as empty.

2.6 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.

2.7 Parameterize the key_i s

`\Ccool<tl2>{<keyval list1>}` is identical to the **Core feature**, except $\langle key_i \rangle$ is replaced by $\langle key_i \langle tl_2 \rangle \rangle$. The default value of $\langle tl_2 \rangle$ is encoded in **Param**. In our example, $\langle tl_2 \rangle$ could be **Style**.

2.8 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**.

`\CcoolClear` `\CcoolClear<tl2>{...<keyi>,...}`

Semantics Clears all $\langle key_i \langle tl_2 \rangle \rangle$'s

`\CcoolHook`

`\CcoolHook`

Semantics No side effect or expansion

`\CcoolLambda`

`\CcoolLambda[⟨arg spec⟩]{⟨code⟩}`,

where *arg spec* is by default an ‘*o*’-type argument.

Example `\Ccool{ EvalAt = \CcoolLambda{(#1)} }`

Semantics Returns a command of type `\DeclareDocumentCommand[5]`,

`\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

Also see **Part IV And**

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

Syntax *keyval list*

Expans

Also see **Core feature** and **Part IV Expans**

Syntax *eo|ee|ex|xoxe|xx*

File

Also see **Part I Write** and **Part IV File**

Syntax *path*

Inner

Also see **Process the val_is** and **Part IV Inner**

Syntax *code*, with **####1** as the *placeholder*

Param

Also see **Parameterize the key_is**, and **Part IV Param**

Syntax *token list*

Outer

Also see **Expand the val_is**, and **Part IV Outer**

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

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

Separ

Also see [Expand the *vals*](#); Listing 8; and [Part IV Separ](#)

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

Syntax That of *separators* in [2, Section 8 of l3seq]

Write

Also see [Part I Write](#) and [Part IV Write](#)

Syntax `\BooleanFalse|\BooleanTrue`

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

Also see [Part IV \CcoolRead](#)

Semantics

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

`\CcoolVers` `\CcoolVers`

Semantics \rightarrow the package’s version

9 Do’s and dont’s

1)

Don’t: `Inner=\{####1\}`

Symptom: `\CcoolRead` fails

Do: `Inner={\char‘{####1\char‘}}`

2)

Don’t: $\langle key_i \rangle <x \mathbb{S}$.

Do: $\langle key_i \rangle \{ < \} x \mathbb{S}$

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: Use '*d*'-type or '*e*'-type arguments[5] for `\CcoolLambda`

Do: Use only '*m*'-type and '*o*'-type arguments

7)

Don't: `\usepackage[spanish]{babel}`

Do: `\usepackage[spanish.noquoting]{babel}[11]`

8) Also see [Part III, section 4](#)

Part II

Listing

NB:

1. Some statements affect only the output of listings that come after that in which they appear. The demarcation is indicated by `%^^A---` and `%^^A<---`, where applicable

Listing 1. `\CcoolVers`

```
\CcoolVers
```

2020/10/10 v3.1 cool — A key-value document command parser

Listing 2. This doc’s dependencies

```
\ExplSyntaxOn
\_ccooldoc_usepackage:
\ExplSyntaxOff
```

```
\usepackage{amsmath,amssymb}
\usepackage[french,german,spanish.noquoting,english]{babel}
\usepackage{bookmark,ccool,microtype,tabto,tcolorbox}
\usepackage[T1]{fontenc}
```

Listing 3. “Let \mathbb{N} and \mathbb{R} denote...” (start of the tutorial)

```
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 4. Equivalent to [3](#), 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 5. Equivalent to [4](#), with `\Ccool`

```
% ^^A--->
\Ccool c{\mathbb{#1}}{ Nat = {N}, Real = {R} }
Let~$\Nat$ and $\Real$~denote the natural and real numbers.
% ^^A<---
\CcoolClear
```

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

Listing 6. Equivalent to 5, with expansion

```
% ^^A--->
\Ccool[Let~]
c{\mathbb{#1}}{ Nat = {N}, Real = {R} }*
[~denote the natural and real numbers.]{ }
% ^^A<---
\CcoolClear
```

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

Listing 7. Equivalent to 5, parameterized (end of the tutorial)

```
% ^^A--->
\Ccool<Style>c{\mathbb{#1}}{ Nat = {N}, Real = {R} }
[Let $\Nat<Style>$ and $\Real<Style>$ denote the natural and real
 numbers.]{ }
% ^^A<---
\CcoolClear<Style>
```

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

Listing 8. Language and mode

```
% ^^A--->
\textbf{\languagename}{:}~\Ccool{ X = x, Y = y }*
\begin{otherlanguage}{spanish}
\CcoolOption[ Separ ]\
\textbf{\languagename}{:}~\Ccool{ X = x, Y = y }*
\end{otherlanguage}\
\textbf{\languagename}{:}~\Ccool{ X = x, Y = y }*
\\[1em]
\CcoolOption[ Outer = ####1 ]
\textbf{\languagename}{:}~\Ccool{ X = this, Y = that }*
\begin{otherlanguage}{spanish}
\CcoolOption[ Separ ]\
\textbf{\languagename}{:}~\Ccool{ X = esto, Y = aquello }*
\end{otherlanguage}\
\textbf{\languagename}{:}~\Ccool{ X = this, Y = that }*
\CcoolOption[ Separ ]\
% ^^A<---
\CcoolOption
```

english: x and y

spanish: x y y

english: x and y

english: this and that
spanish: esto y aquello
english: this and that

Listing 9. Separators (*Note*^a)

^a[bug]: Removing the closing `\CcoolOption` subsequently causes inconsistent separators between text and math mode (case replicated in uncommented form in `dtx`)

```

% ^^A--->
\CcoolOption[ Separ={{\ \char`@ }{\ \% }{\ \char`@ } } ]
\Ccool{ X = x, Y = y }*[\]
{ X = x, Y = y }*s{{~\&~}}[\]
{ X = x, Y = y }*s{{,~}{~\&~}}[\][1em]
{ X = x, Y = y, Z = z }*[\]
{ X = x, Y = y, Z = z }*s{{~\&~}}[\]
{ X = x, Y = y, Z = z }*s{{,~}{\&~}}[\]
{ X = x, Y = y, Z = z }*s{{~\&~}{,~}{,~\&~}}\
% ^^A<---
\CcoolOption
\CcoolClear

```

```

x @ y
x & y
x & y

x % y @ z
x & y & z
x, y, & z
x, y, & z

```

Listing 10. Hello, world! (testing)

```

\CcoolOption[ Write = \BooleanTrue ]
% ^^A--->
\CcoolOption[Separ = {{}{.}{.}}, Outer = {####1}]
\Ccool
<Test>{ KeyA = {.,} , KeyB = {!}, KeyC = {\%} }[]
<Test>{ KeyD = {d}, KeyE = {\%} }[]
<Test>c{\#1\}{ KeyF = {H}, KeyG = {e}, KeyH = {1} }*[]
<Test>{ KeyI = {\%}, KeyJ = {\%}, KeyK = {\%} }[.\{1\}.\{o\}]
<Test>{ KeyL = {1}, KeyM = {\char`[]}, KeyN = {\char`[] } }[]
<Test>{ KeyO = {o}, KeyP = {\%}, KeyQ = {\%} }[{\, \ ]}
<Test>{ KeyR = {w}, KeyS = {o}, KeyT = {r} }*
s{{}{}}c{{\char`[]#1}[]
<Test>{ KeyU = {\%}, KeyV = {\%}, KeyW = {\%} }[]
<Test>{ KeyX = {\%}, KeyY = {\%}, KeyZ = {KeyB<Test>} }\nobreak
\KeyL<Test>\KeyD<Test>\KeyZ<Test>\KeyN<Test>\
% ^^A<---

```

```
\CcoolOption
\CcoolClear
```

{H}.{e}.{l}.{l}.{o}, [world!]

Listing 11. Listing 10 read from file

```
% ^^A--->
\CcoolRead
\KeyF<Test>\KeyA<Test>\nobreak
\KeyG<Test>\KeyA<Test>\nobreak
\KeyH<Test>\KeyA<Test>\nobreak
\KeyH<Test>\KeyA<Test>\nobreak
\KeyH<Test>\KeyA<Test>\nobreak
\KeyH<Test>\KeyA<Test>\nobreak
\KeyM<Test>\KeyR<Test>\nobreak
\KeyO<Test>\nobreak
\KeyT<Test>\nobreak
\KeyL<Test>\nobreak
\KeyD<Test>\nobreak
\KeyZ<Test>\nobreak
\KeyN<Test>\nobreak
% ^^A<---
\CcoolClear
```

{H}.{e}.{l}.{l}.{o}, [world!]

Listing 12. Probability space

```
\CcoolOption[ Write = \BooleanTrue ]
% ^^A--->
\Ccool[Let~]
{ Space = \Omega, Field = \mathcal{F}, Meas = \mathcal{P} }
*s{,{,}c{${#\1}$}
[~denote the probability space, where~]{ PowerSet = { 2^{\Space} } }
[{\Field\subset \PowerSet$.]
{}
% ^^A<---
\CcoolClear
\CcoolOption
```

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

Listing 13. Listing 12 read from file

```
% ^^A--->
\CcoolRead \tab $\Omega$ $\Field$ $\Meas$
% ^^A<---
\CcoolClear
```

Listing 14. Mittelwertsatz für n Variable[4, 17.3]

```

\CoolOption[ Write = \BooleanTrue ]
% ^^A--->
\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] } }+
[$n\in\mathbb{N}$,~$\text{OffMenge}\subseteq\mathbb{R}^n$ 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: $\mathbb{N}$, $\xi$)
% ^^A<---
\CoolClear
\CoolOption

```

Theorem 1 (Mittelwertsatz für n Variable) *Es sei $n \in \mathbb{N}$, $D \subseteq \mathbb{R}^n$ 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 15. Listing 14 read from file

```

% ^^A--->
\CoolRead \tab $\mathbb{N}$ $\mathbb{R}$ $\text{OffMenge}$ $\text{Ci}$ $\text{Strecke}$
% ^^A<---
\CoolClear

```

Listing 16. Families of polynomial functions

```

\CoolOption[ Write = \BooleanTrue ]
% ^^A--->
\Cool c{\mathbb{#1}}{ Nat = {N}, Real = {R} }
[Let~]
{ PolyR = \CoolLambda[o]{\Real\IfValueT{#1}{_#1}[X] } }
[ $\text{PolyR}[n]$  et  $\text{PolyR}$ , denote the families of polynomial functions
on  $\mathbb{R}$ , of order  $n$  et and their union over  $n \in \mathbb{N}$ ,
respectively. ]
{}
% ^^A<---
\CoolClear
\CoolOption

```

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 17. Listing 16 read from file

```

% ^^A--->
\CoolRead \tab  $\text{PolyR}[n]$  et  $\text{PolyR}$ 
% ^^A<---
\CoolClear

```

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

Listing 18. Same as Listing 16, but arbitrary number system

```

\CoolOption[ Write = \BooleanTrue ]
% ^^A--->
\selectlanguage{french}
\Cool c{\mathbb{#1}}{ Corps = {K}, Nat = {N}, Reel = {R} }
[Soient~]
{
  Poly = \CoolLambda[om]{#2\IfValueT{#1}{_#1}[X] },
  PolyR = \CoolLambda[o]{\Poly[#1]{\Reel}}
}
[ $\text{Poly}[n]{\text{Corps}}$  et  $\text{Poly}{\text{Corps}}$ , les familles de polynômes sur
 $\text{Corps}$ , de degré  $n$  et leur union pour  $n \in \mathbb{N}$ ,
respectivement. En particulier,
ils sont d'notés  $\text{PolyR}[n]$  et  $\text{PolyR}$ , pour  $\text{Corps}=\mathbb{R}$ .]
{}
% ^^A<---
\CoolClear
\CoolOption

```

Soient $\mathbb{K}_n[X]$ et $\mathbb{K}[X]$, les familles de polynômes sur \mathbb{K} , de degré n et leur union pour $n \in \mathbb{N}$, respectivement. En particulier, ils sont dénotés $\mathbb{R}_n[X]$ et $\mathbb{R}[X]$, pour $\mathbb{K} = \mathbb{R}$.

Listing 19. Listing 18 read from file

```
% ^^A--->
\CcoolRead \tab $\PolyR[n]$ et $\PolyR$
% ^^A<---
\CcoolClear
```

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

Listing 20. Fonction et fonctionnelle

```
\CcoolOption[ Write = \BooleanTrue ]
% ^^A--->
\selectlanguage{french}
\Ccool{ EvalAt = \CcoolLambda{(#1)}, ApplyOp = \CcoolLambda[mm]{#1[#2]} }
[Supposons une fonction $f\EvalAt{t}$, et \etudions le probl\eme o\`u
 la fonctionnelle $\ApplyOp{S}{f}$ est donn\ee par\dots]{}
% ^^A<---
\CcoolClear
\CcoolOption
```

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

Listing 21. Listing 20 read from file

```
% ^^A--->
\CcoolRead \tab $f\EvalAt{t}$, $\ApplyOp{S}{f}$
% ^^A<---
\CcoolClear
```

$f(t), S[f]$

Listing 22. CUSUM statistic 

```
\CcoolOption[ Write = \BooleanTrue ]
% ^^A--->
\newtheorem{definition}{Definition}
\AfterEndEnvironment{definition}{\CcoolHook}
\Ccool{
  SuchThat = { ;~ },
  Time = { t },
  Process = { \xi },
  StopT = { T },
  EvalAt = \CcoolLambda{(#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: $\Scale$, $\CUSUM$)
% ^A<---
\CoolClear
\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 23. Listing 22 read from file

```
% ^^A--->
\CcoolRead \tab $\Time $ $\Process$ $\Scale$ $\Real$ $\CUSUMthresh$
  $\LogWald$ $\CUSUMst$ $\CUSUM$ $\LogWaldInf$
% ^^A<---
\CcoolClear
```

t ξ λ R ν u T_c y m

Part III

Other

1 Acknowledgment

This work has benefited from Q&A's from the L^AT_EXcommunity[7][10]. Specific attributions are made throughout this document.

2 Genealogy

“Give commands the ability to contain the mathematical meaning while retaining the typesetting versatility” (cool[1]). The addition of ‘c’, in ccool, is for *custom*. With hindsight it is restrictive to describe ccool as a tool for encoding mathematical convention.

3 Install

- 1) Compile ccool.dtx (under Unix, `$pdflatex ccool.dtx`)
- 2) Put the generated ccool.sty in the search path of the L^AT_EXengine

4 Issue

Listed under:

- a) NOTE or \NB, tagged either `bug` or `fixed`, inside ccool.dtx

5 Support

This package is available from <https://www.ctan.org/pkg/ccool> and <https://github.com/rogard/ccool>.

6 Testing

6.1 Technicality

Not possible to compile-check the expansion of a certain class of macros against predefined values[8]. Instead, one can

- a) Follow the steps in [section 3](#) on one's own machine to generate ccool.pdf
- b) Visually check [Part II](#), against that [of the repository](#), for the same version.

Also see:

- c) Also see: [section 7 a\)](#)

6.2 Platform

- i)* Linux laptop 4.15.0-20-generic #21-Ubuntu SMP Tue Apr 24
↔ 06:16:15 UTC 2018 x86_64 x86_64 x86_64 GNU/Linux

6.3 Engine

- a)* pdfTeX 3.14159265-2.6-1.40.20 (TeX Live 2019)
- b)* pdfTeX 3.14159265-2.6-1.40.21 (TeX Live 2020)
- c)* LuaHBTeX, Version 1.12.0 (TeX Live 2020)
- d)* XeTeX 3.14159265-2.6-0.999992 (TeX Live 2020)

6.4 Results

- 1) ccool v1.8 compiles satisfactorily on platform *i)* and engine *a)*
- 2) ccool v1.8 compiles satisfactorily on platform *i)* and engine *b)*
- 3) ccool v1.9 compiles satisfactorily on platform *i)* and engines *b)* and *c)*
- 4) ccool v2.0 compiles satisfactorily on platform *i)* and engines *b)*, *c)*, and *d)*
- 5) ccool v2.1 compiles satisfactorily on platform *i)* and engines *b)*, *c)*, and *d)*
- 6) ccool v2.3 compiles satisfactorily on platform *i)* and engines *b)*, *c)*, and *d)*
- 7) ccool v2.7 compiles satisfactorily on platform *i)* and engines *b)*, *c)*, and *d)*
- 8) ccool v2.8 compiles satisfactorily on platform *i)* and engines *b)*, *c)*, and *d)*

6.5 Other

Check [6] for testing ccool with llncs

7 To do

- a)* Regression testing using [3, Section 3.2—Specifying expectations].

Also see:

- b)* NOTE or \NB tagged `abandon|done|todo` inside `ccool.dtx`

References

- [1] Nick Setzer *The cool package*, 2005, <https://www.ctan.org/pkg/cool>
- [2] The L^AT_EX3 Project Team *The L^AT_EX3 interfaces*, 2019, <http://ftp.math.purdue.edu/mirrors/ctan.org/macros/latex/contrib/l3kernel/interface3.pdf>
- [3] The L^AT_EX3 Project Team *The l3build package*, 2020, <http://mirror.utexas.edu/ctan/macros/latex/contrib/l3build/l3build.pdf>
- [4] Thomas F. Sturm *The tcolorbox package*, 2019, <http://www.texdoc.net/texmf-dist/doc/latex/tcolorbox/tcolorbox.pdf>
- [5] The L^AT_EX3 Project Team *The xparse package*, 2020, <http://ftp.math.purdue.edu/mirrors/ctan.org/macros/latex/contrib/l3packages/xparse.pdf>
- [6] Erwann Rogard and Olympia Hadjiliadis *Typesetting a math thesis with ccool*, 2020, <https://github.com/rogard/ccool/blob/master/thesis.pdf>
- [7] <https://tex.stackexchange.com/users/112708/erwann?tab=questions>
- [8] @joseph-wright’s answer to “Checking a function’s expansion against a string”, <https://tex.stackexchange.com/a/534100>
- [9] @frougon’s answer to “Journaling calls to a function []”, <https://tex.stackexchange.com/a/536620>
- [10] \Ccool, extension à L^AT_EX à vocation mathématique, <http://forum.mathematex.net/latex-f6/ccool-extension-latex-a-vocation-mathematique-t17314.html>
- [11] @Javier Bezos’s answer to <https://tex.stackexchange.com/a/547018/112708>

Change History

v1.0		recommended[5] 21
General: Initial version	21	Replace: <code>GenericObject</code> by <code>Name</code> 21
v1.1		Replace: <code>Separators</code> by <code>Separ</code> 21
General: Add: <code>Save</code>	21	v1.2
Add: Listing 1., 2., 3., 4., 6., and 9.	21	General: Add: optional <code>*to \OpsNew</code>
Add: <code>\OpsRestore</code>	21	as instruction to expand keyval list ₁ 21
Add: <code>\OpsTest</code>	21	Delete: <code>\OpsTest</code> 21
Delete: Listing 1-5 from v1.0	21	Delete: <code>\keyval2</code> and <code>\code2</code> 21
Fix: apparent anomaly in v1.0’s		Delete: Listing 2-3 from v1.1. 21
Listing 4, see Listing 10	21	Replace: <code>\OpsClear{\tl2}</code> by
Rearrange: much of the		<code>\OpsClear[\keyval list]</code> 21
implementation	21	Replace: <code>\Restore</code> by <code>\Read</code> 21
Replace:		Replace: <code>\Save</code> by <code>\Write</code> 21
<code>\OpsOptions</code> by <code>\OpsOption</code>	21	v1.3
Replace: <code>{\keyval2}</code> by <code><\keyval2></code> given		General: Replace: <code>\OpsNew</code> by <code>\Ops</code> 21
that option type <code>G</code> not		

	Replace: $\langle tl_2 \rangle$ and $[tl_2]$ by $\langle tl_2 \rangle$	21	v2.1	General: Add: Listings 3, 4, 5, 6, 7, 8, and 9	21
v1.4	General: Add: section 9	21		Replace: $\langle tl_2 \rangle$'s position within $\backslash\text{Ccool}$'s argument list, from first to second. Greater versatility	21
	Add: $\backslash\text{OpsDebug}$	21		Replace: $\backslash\text{CcoolLambda}$'s optional integer argument (number of m's) by a standard argument list	21
	Add: $\backslash\text{OpsHook}$	21		Replace: <i>global option Name</i> by Param	21
	Add: Expans (for debugging' sake, but...)	21		Replace: as the default of Param , Math by Default	21
	Add: Listing 1., 2., and 3.	21			
	Add: optional +to $\backslash\text{OpsNew}$ to make side effects persist beyond local group	21	v2.2	General: Remove: % from listings	21
	Delete: Listing 1., and 2.	21		Replace: part of the abstract's with more straightforward descriptions based on input from forum participants	21
	Replace: $\text{s}\{\langle tl_3 \rangle\}\{\langle tl_4 \rangle\}\{\langle tl_5 \rangle\}$ by $\text{s}\{\langle tl_3 \rangle\}\{\langle tl_3 \rangle\}\{\langle tl_4 \rangle\}\{\langle tl_3 \rangle\}\{\langle tl_4 \rangle\}\{\langle tl_5 \rangle\}$	21			
v1.5	General: Add: File	21			
	Delete: dependence on datetime	21	v2.3	General: Add: Listing 17 , Listing 18 , and Listing 19	21
v1.6	General: Add: Listing showing part of the preamble	21		Complete: Listing 16	21
	Rename: $\backslash\text{OpsClear}$ to $\backslash\text{CcoolClear}$	21		Rearranged: $\backslash\text{Ccool}$'s subsections. Previously, by argument. Now, by feature.	21
	Rename: $\backslash\text{OpsDebug}$ to $\backslash\text{CcoolDebug}$	21		Remove: Listing showing part of the preamble	21
	Rename: $\backslash\text{OpsHook}$ to $\backslash\text{CcoolHook}$	21		Replace: for $\backslash\text{Ccool}$, $\text{i}\{\}$ by $\text{c}\{\}$	21
	Rename: $\backslash\text{OpsOption}$ to $\backslash\text{CcoolOption}$	21		Replace: In step 2) , the created command's implementation, from $\backslash\text{ProvideDocumentCommand}$ to $\backslash\text{DeclareDocumentCommand}$	21
	Rename: $\backslash\text{OpsRead}$ to $\backslash\text{CcoolRead}$	21	v2.4	General: Fix: minor error in the listings ($\backslash\text{Real}$ rather than $\backslash\text{Reel}$, hitherto unnoticed).	21
	Rename: $\backslash\text{Ops}$ to $\backslash\text{Ccool}$	21		Remove: examples from Part I , $\backslash\text{Ccool}$, as redundant with Part II Listing 3-7	21
	Rename: oops to ccool (better describes the purpose)	21			
v1.7	General: Add: Legends to listings	21			
	Add: Listing 22 (CUSUM)	21			
	Delete: $\backslash\text{CcoolDebug}$	21			
	Delete: Listing 5 from v1.6	21			
v1.8	General: Add: $\backslash\text{CcoolVers}$	21	v2.5	General: Modify: File , rely on erw-l3 's $\backslash\text{erw_jobnametimestamp}$:	21
	Add: $\backslash\text{CcoolLambda}$	21		Modify: behavior of Part I Expand the <i>vals</i> , rely on erw-l3 's $\backslash\text{erw_seq_use}$: Nn	21
	Add: Listing 20 , Listing 21	21			
	Add: Listing 1	21	v2.6	General: Modify: $\backslash\text{CcoolLambda}$, rely on erw-l3 's $\backslash\text{erw_lambda:nnn}$	21
v1.9	General: Add: support for Lua \TeX	21	v2.7	General: Add: <i>global option And</i>	21
	Move: from Part I to Part IV , what is now that part's section 11	21		Add: Listing 8	21
v2.0	General: Add: support for X \TeX	21		Modify: $\backslash\text{CcoolOption}$'s 'm'-type argument by 'o'-type argument	21
	Delete: File 's dependency on texosquery and $\backslash\text{pdfcreationdate}$	21			
	Update: $\backslash\text{RequirePackage}$, $\backslash\text{NeedsTeXFormat}$'s second argument / TeX Live 2020	21			

Modify: Separ 's default rely on	v3.0
babel and amsmath , if applicable	General: Miscellaneous 21
v2.8	v3.1
General: Fix: conflict between	General: Replaced: Listing 2's content,
\usepackage[spanish]{babel} and	from exhaustive dependencies to
Parameterize the <i>key</i>s	those explicit with \usepackage 21
v2.9	
General: Miscellaneous	

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	
\'	103
\(key_i<tl_2>)	6
\(key_i)	5, 8
\(key_i)	5
And (option)	7, 36
Expans (option)	7, 36
File (option)	7, 37
Inner (option)	7, 37
Outer (option)	7, 37
Param (option)	7, 37
Separ (option)	7, 38
Write (option)	7, 38
\^	108
_	273, 274, 275, 281, 282, 283
A	
\AtEndDocument	125
B	
\begingroup	210
bool commands:	
\bool_gset_false:N	130
\bool_gset_true:N	137
\bool_if:nTF	152, 206, 229, 415
\bool_set_false:N	126
\BooleanFalse	8, 420, 421
\BooleanTrue	6, 8
C	
\c	115
\Ccool	1, 2, 5, 6, 6, 7, 10, 21, 22, 211, 218, 239
ccool internal commands:	
__ccool_aux_inner:n	6, 7, 38
__ccool_aux_inner_set:n	4, 196
__ccool_aux_key:N	17, 200
__ccool_aux_key:n	13, 20
__ccool_aux_key:w	9, 15
\g_ccool_aux_key_seq	11, 19, 181, 201, 305, 306
\g_ccool_aux_keyval_seq	197, 198, 200
__ccool_aux_outer:n	24, 183
__ccool_aux_outer_set:n	22, 182
\g_ccool_aux_prop	26, 29, 46, 199
__ccool_aux_prop:N	44, 198
__ccool_aux_prop:n	40, 50
__ccool_aux_prop:nn	26
__ccool_aux_prop:w	32, 42
__ccool_aux_val:Nn	53, 181
\g_ccool_aux_val_seq	55, 56, 187
__ccool_lang_and:	68, 273, 275, 281, 283
__ccool_lang_and:n	68
\g_ccool_lang_and_prop	58, 62, 71, 73
\c_ccool_lang_and_tl	86, 347, 348
__ccool_lang_and_update:n	59, 346
__ccool_log_close:	124, 417
__ccool_log_entry	211, 212
\g_ccool_log_file_tl	132, 135, 354
\g_ccool_log_iow	124, 125, 129, 136, 154, 157
__ccool_log_open:	132, 416
\g_ccool_log_open_bool	126, 130, 137, 152, 206
__ccool_log_read:	145, 428
__ccool_log_read:n	139, 147, 427
\g_ccool_log_to_tl	135, 136, 147, 149
__ccool_log_write:n	149, 208
__ccool_make_ccool:nnnn	216, 361, 375, 389, 403
__ccool_make_ccool_exp:nnn	179, 227
__ccool_make_ccool_key:nnn	191, 205
__ccool_make_ccool_sideeffect:nnn	203, 224, 233

<code>__ccool_make_key:N</code>	175, 201
<code>__ccool_make_key:n</code>	170, 177
<code>__ccool_make_key:Nn</code>	160, 172
<code>\g__ccool_option_expans_tl</code> 32, 36, 350	
<code>__ccool_option_inner:n</code>	249, 359
<code>\g__ccool_option_inner_tl</code>	
.	249, 252, 364, 378, 392, 406
<code>__ccool_option_outer:n</code>	259, 387
<code>\g__ccool_option_outer_tl</code>	
.	259, 262, 366, 380, 394, 408
<code>__ccool_option_param:n</code>	254, 373
<code>\g__ccool_option_param_tl</code>	
.	164, 254, 257, 330, 363, 377, 391, 405
<code>__ccool_option_separ:n</code>	264, 401
<code>\c__ccool_option_separ_default_</code> <code>tl</code>	271, 279, 412, 413
<code>\g__ccool_option_separ_tl</code>	
.	264, 267, 269, 365, 379, 393, 407
<code>__ccool_prop_append:NN</code>	286, 297
<code>__ccool_prop_append:Nn</code>	199, 295
<code>__ccool_prop_append:nn</code>	288, 292
<code>__ccool_prop_clear_new:n</code>	299, 306
<code>__ccool_prop_clear_new_map:n</code>	
.	303, 332
<code>__ccool_prop_if_exist:nTF</code>	193, 308
<code>__ccool_prop_item:nn</code>	166, 312
<code>__ccool_prop_name:n</code>	
.	56, 297, 301, 310, 314, 316, 319
<code>__ccool_prop_new:n</code>	195, 317
<code>__ccool_seq_from_prop:n</code>	323, 327
<code>__ccool_seq_from_prop:NNn</code>	56, 321
<code>\CcoolClear</code>	2, 4, 6, 22, 329
<code>\CcoolDebug</code>	22
<code>\CcoolHook</code>	2, 4, 5, 6, 22, 231, 334
<code>\CcoolLambda</code>	2, 4, 5, 6, 8, 22, 335
<code>\CcoolOption</code>	2, 4, 7, 22, 339
<code>\CcoolRead</code>	2, 4, 8, 8, 22, 423
<code>\CcoolVers</code>	2, 4, 8, 10, 22, 430
cs commands:	
<code>\cs_generate_variant:Nn</code>	
.	7, 31, 67, 144, 159, 169, 174, 294
<code>\cs_gset:Npn</code>	6, 24, 267
<code>\cs_new:Nn</code>	68, 81, 84, 308, 312
<code>\cs_new:Npn</code>	316
<code>\cs_new_protected:Nn</code>	
.	4, 13, 17, 22, 27, 40,
.	44, 53, 59, 127, 133, 139, 145, 150,
.	160, 170, 175, 179, 191, 203, 250,
.	255, 260, 265, 295, 299, 303, 317, 321
<code>\cs_new_protected:Npn</code>	9, 33, 216, 286
<code>\cs_set:Nn</code>	288
<code>\cs_set_protected:Nn</code>	323
	D
<code>\DeclareDocumentCommand</code> 6, 22, 163, 218, 337	
<code>\def</code>	211
	E
<code>\endgroup</code>	212
<code>\ensuremath</code>	398, 399
erw commands:	
<code>\erw_jobnametimestamp:</code>	22
<code>\erw_keyval_error:Nn</code>	63
<code>\erw_lambda:nnn</code>	22, 337
<code>\erw_prop_keyval_parse:NNNn</code>	61
<code>\erw_seq_use:Nn</code>	22, 186
<code>\erw_sys_jobnametimestamp:</code>	356, 357
exp commands:	
<code>\exp_args:NNf</code>	185
<code>\exp_args:NNx</code>	162, 218
<code>\exp_args:No</code>	81, 196, 301
<code>\exp_args:Nx</code>	35
<code>\exp_last_unbraced:Nf</code> 360, 374, 388, 402	
<code>\exp_not:n</code>	239
<code>\expandafter</code>	211
<code>\ExplSyntaxOff</code>	433
<code>\ExplSyntaxOn</code>	3
	F
file commands:	
<code>\file_input:n</code>	141
<code>\function</code>	5
	G
<code>\gappto</code>	231
	I
<code>\IfBooleanT</code>	225
<code>\ifcsdef</code>	79, 269
<code>\IfValueT</code>	223, 237
<code>\IfValueTF</code>	426
iow commands:	
<code>\iow_close:N</code>	125, 129
<code>\iow_new:N</code>	124
<code>\iow_now:Nn</code>	154
<code>\iow_open:Nn</code>	136
	K
keys commands:	
<code>\l_keys_choice_tl</code>	350
<code>\keys_define:nn</code>	344
<code>\keys_set:nn</code>	342
	L
<code>\languagename</code>	81
	M
msg commands:	
<code>\msg_error:nnn</code>	157

<code>\msg_new:nnn</code>	243, 246		
<code>\msg_warning:nnn</code>	75		
N			
<code>\NB</code>	19, 20		
<code>\NeedsTeXFormat</code>	22		
<code>\NewDocumentCommand</code>	2, 10, 329, 334, 339, 423, 430		
O			
<code>\Ops</code>	21, 22		
<code>\OpsClear</code>	21, 22		
<code>\OpsDebug</code>	22		
<code>\OpsHook</code>	22		
<code>\OpsNew</code>	21, 22		
<code>\OpsOption</code>	21, 22		
<code>\OpsOptions</code>	21		
<code>\OpsRead</code>	22		
<code>\OpsRestore</code>	21		
<code>\OpsTest</code>	21		
options:			
And	7, 36		
Expans	7, 36		
File	7, 37		
Inner	7, 37		
Outer	7, 37		
Param	7, 37		
Separ	7, 38		
Write	7, 38		
P			
<code>\pdfcreationdate</code>	22		
prop commands:			
<code>\prop_clear_new:N</code>	301		
<code>\prop_gclear_new:N</code>	46		
<code>\prop_gput:Nnn</code>	29, 64, 290		
<code>\prop_if_exist:NTF</code>	310		
<code>\prop_if_in:NnTF</code>	70		
<code>\prop_item:Nn</code>	73, 290, 314, 325		
<code>\prop_map_function:NN</code>	292		
<code>\prop_new:N</code>	26, 58, 319		
<code>\ProvideDocumentCommand</code>	22, 335		
Q			
quark commands:			
<code>\q_stop</code>	9, 15, 33, 42		
R			
<code>\Read</code>	21		
<code>\Real</code>	1, 5, 22		
<code>\Reel</code>	1, 22		
<code>\RequirePackage</code>	22		
<code>\Restore</code>	21		
S			
<code>\Save</code>	21		
seq commands:			
<code>\seq_gclear_new:N</code>	19, 55		
<code>\seq_gput_right:Nn</code>	11, 325		
<code>\seq_if_empty:NTF</code>	47		
<code>\seq_map_function:NN</code>	20, 50, 177, 306, 327		
<code>\seq_set_from_clist:Nn</code>	197, 305		
T			
<code>\text</code>	273, 274, 275		
tl commands:			
<code>\c_empty_tl</code>	48, 194, 214, 236, 334		
<code>\tl_const:Nn</code>	86, 271, 279		
<code>\tl_gset:Nn</code>	135, 252, 257, 262, 354		
<code>\tl_gset_eq:NN</code>	350		
<code>\tl_log:n</code>	142, 155		
<code>\tl_new:N</code>	32, 132, 149, 249, 254, 259, 264		
<code>\tl_trim_spaces:n</code>	11, 37, 38		
U			
use commands:			
<code>\use:N</code>	36, 432		
<code>\usepackage</code>	5, 23		
W			
<code>\Write</code>	21		
X			
<code>\X</code>	5		

Part IV

Implementation

1 Opening

```
1 <*package>
2 <@@=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_inner_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`.)

3 lang

```
58 \prop_new:N \g__ccool_lang_and_prop
\__ccool_lang_and_update:n
59 \cs_new_protected:Nn \__ccool_lang_and_update:n
60 {
61   \erw_prop_keyval_parse:NNNn
62   \g__ccool_lang_and_prop
63   \erw_keyval_error:Nn
64   \prop_gput:Nnn
65   { #1 }
66 }
67 \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:
68 \cs_new:Nn \__ccool_lang_and:n
69 {
70   \prop_if_in:NnTF
71   \g__ccool_lang_and_prop
72   {#1}
73   {\prop_item:Nn\g__ccool_lang_and_prop{#1}}
74   {
75     \msg_warning:nnn{__ccool}{lang_and}{#1}
76     \__ccool_lang_and:n{english}
77   }
78 }
79 \ifcsdef{language}
80 {
81   \cs_new:Nn \__ccool_lang_and:{\exp_args:No\__ccool_lang_and:n{language}}
82 }
83 {
84   \cs_new:Nn \__ccool_lang_and:{english}
85 }

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

\c__ccool_lang_and_tl (Note1)
86 \tl_const:Nn \c__ccool_lang_and_tl
87 {
88   %^^A https://www.overleaf.com/learn/latex/International\_language\_support
89   afrikaans=en,
90   basque=eta,
91   catalan=i,
92   croatian=i,
93   czech=a,
94   danish=og,
95   dutch=en,
96   english=and,
97   esperanto=kaj,
98   estonian=ja,
```

¹[todo]: Non latin-alphabet languages

```

99   finnish=ja,
100  french=et,
101  galician=e,
102  german=und,
103  hungarian=\`es,
104  icelandic=og,
105  indonesian=dan,
106  irish=agus,
107  italian=e,
108  kurmanji=\^u,
109  latin=et,
110  latvian=un,
111  lithuanian=ir,
112  ngerman=und,
113  polish=i,
114  portuguese=e,
115  romanian=\c{s}i,
116  slovak=a,
117  spanish=y,
118  swedish=och,
119  swissgerman=und,
120  turkish=ve,
121  turkmen=we,
122  welsh=a
123 }

```

(End definition for \c__ccool_lang_and_tl.)

4 log

__ccool_log_close:

```

124 \iow_new:N \g__ccool_log_iow
125 \AtEndDocument{\iow_close:N \g__ccool_log_iow}
126 \bool_set_false:N \g__ccool_log_open_bool
127 \cs_new_protected:Nn \__ccool_log_close:
128 {
129   \iow_close:N \g__ccool_log_iow
130   \bool_gset_false:N \g__ccool_log_open_bool
131 }

```

(End definition for __ccool_log_close:.)

__ccool_log_open:

```

132 \tl_new:N \g__ccool_log_file_tl
133 \cs_new_protected:Nn \__ccool_log_open:
134 {
135   \tl_gset:Nx \g__ccool_log_to_tl{\g__ccool_log_file_tl}
136   \iow_open:Nn \g__ccool_log_iow {\g__ccool_log_to_tl}
137   \bool_gset_true:N \g__ccool_log_open_bool
138 }

```

(End definition for __ccool_log_open:.)

```

\__ccool_log_read:n #1 : <path>
139 \cs_new_protected:Nn \__ccool_log_read:n
140 {
141   \file_input:n{#1}
142   \tl_log:n{read~from~#1}
143 }
144 \cs_generate_variant:Nn \__ccool_log_read:n { e }
(End definition for \__ccool_log_read:n.)

```

```

\__ccool_log_read:
145 \cs_new_protected:Nn \__ccool_log_read:
146 {
147   \__ccool_log_read:e{\g__ccool_log_to_tl}
148 }
(End definition for \__ccool_log_read:.)

```

```

\__ccool_log_write:n
149 \tl_new:N \g__ccool_log_to_tl
150 \cs_new_protected:Nn \__ccool_log_write:n
151 {
152   \bool_if:nTF{ \g__ccool_log_open_bool }
153   {
154     \iow_now:Nn \g__ccool_log_iow {#1}
155     \tl_log:n{ write-to~#1 }
156   }
157   { \msg_error:nnn{ __ccool }{ iow }{ \g__ccool_log_iow } }
158 }
159 \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 >
160 \cs_new_protected:Nn \__ccool_make_key:Nn
161 {
162   \exp_args:NNx
163   \DeclareDocumentCommand{#1}
164   { D<>{\g__ccool_option_param_tl} }
165   {
166     \__ccool_prop_item:nn{##1}{#2}
167   }
168 }
169 \cs_generate_variant:Nn \__ccool_make_key:Nn { c }
(End definition for \__ccool_make_key:Nn.)

```

```

\__ccool_make_key:n #1: < key >
170 \cs_new_protected:Nn \__ccool_make_key:n
171 {
172   \__ccool_make_key:cn{#1}{#1}
173 }
174 \cs_generate_variant:Nn \__ccool_make_key:n { e }

(End definition for \__ccool_make_key:n.)

```

```

\__ccool_make_key:N #1: < seq >
175 \cs_new_protected:Nn \__ccool_make_key:N
176 {
177   \seq_map_function:NN #1 \__ccool_make_key:e
178 }

(End definition for \__ccool_make_key:N.)

```

6 make_ccool

```

\__ccool_make_ccool_exp:nnn
179 \cs_new_protected:Nn \__ccool_make_ccool_exp:nnn
180 {
181   \__ccool_aux_val:Nn \g__ccool_aux_key_seq {#1}
182   \__ccool_aux_outer_set:n{#3}
183   \__ccool_aux_outer:n
184   {
185     \exp_args:NNf
186     \erw_seq_use:Nn
187     \g__ccool_aux_val_seq
188     {#2}
189   }
190 }

(End definition for \__ccool_make_ccool_exp:nnn.)

```

```

\__ccool_make_ccool_key:nnn
191 \cs_new_protected:Nn \__ccool_make_ccool_key:nnn
192 {
193   \__ccool_prop_if_exist:nTF{#1}
194   { \c_empty_tl }
195   { \__ccool_prop_new:n{#1} }
196   \exp_args:No \__ccool_aux_inner_set:n{#2}
197   \seq_set_from_clist:Nn \g__ccool_aux_keyval_seq {#3}
198   \__ccool_aux_prop:N \g__ccool_aux_keyval_seq
199   \__ccool_prop_append:Nn \g__ccool_aux_prop {#1}
200   \__ccool_aux_key:N \g__ccool_aux_keyval_seq
201   \__ccool_make_key:N \g__ccool_aux_key_seq
202 }

(End definition for \__ccool_make_ccool_key:nnn.)

```

```

\__ccool_make_ccool_sideeffect:nnn [9]
203 \cs_new_protected:Nn \__ccool_make_ccool_sideeffect:nnn
204 {
205   \__ccool_make_ccool_key:nnn{#1}{#2}{#3}
206   \bool_if:nTF{ \g__ccool_log_open_bool }
207   {
208     \__ccool_log_write:n
209     {
210       \begingroup
211       \def \__ccool_log_entry { \Ccool<#1>c{#2}{#3} } \expandafter
212       \endgroup \__ccool_log_entry
213     }
214   }\c_empty_tl}
215 }

```

(End definition for __ccool_make_ccool_sideeffect:nnn.)

```

\__ccool_make_ccool:nnnn #1 : < token list >
#2 : < seq1 >
#3 : < seq2 >
#4 : < prop >
216 \cs_new_protected:Npn \__ccool_make_ccool:nnnn #1 #2 #3 #4
217 {
218   \exp_args:NNx \DeclareDocumentCommand \Ccool
219   {%^A 2 3 4 5 6 7 8 9
220   +o D<>{#1} E{ c }{{#2}} m t+ s E{ s c }{{#3}{#4}} +o
221   }
222   {
223     \IfValueT{##1}{##1}
224     \__ccool_make_ccool_sideeffect:nnn{##2}{##3}{##4}
225     \IfBooleanT{##6}
226     {
227       \__ccool_make_ccool_exp:nnn{##2}{##7}{##8}
228     }
229     \bool_if:nTF{##5}
230     {
231       \gappto{\CcoolHook}
232       {
233         \__ccool_make_ccool_sideeffect:nnn{##2}{##3}{##4}
234       }
235     }
236     {\c_empty_tl}
237     \IfValueT{##9}
238     {
239       \exp_not:n{ \Ccool[##9] }
240     }
241   }
242 }

```

(End definition for __ccool_make_ccool:nnnn.)

7 msg


```

243 \msg_new:nnn {__ccool}
244 { iow }
245 {#1~is~closed~can't~write}
246 \msg_new:nnn {__ccool}
247 {lang_and}
248 {~key~#1~missing~for~global~option~'And';~falling~back~on~'english'}

```

8 option

```

\__ccool_option_inner:n #1: <code>
249 \tl_new:N \g__ccool_option_inner_tl
250 \cs_new_protected:Nn \__ccool_option_inner:n
251 {
252   \tl_gset:Nn \g__ccool_option_inner_tl {#1}
253 }

```

(End definition for __ccool_option_inner:n.)

```

\__ccool_option_param:n #1: <token list>
254 \tl_new:N \g__ccool_option_param_tl
255 \cs_new_protected:Nn \__ccool_option_param:n
256 {
257   \tl_gset:Nn \g__ccool_option_param_tl{#1}
258 }

```

(End definition for __ccool_option_param:n.)

```

\__ccool_option_outer:n #1: < inline code >
259 \tl_new:N \g__ccool_option_outer_tl
260 \cs_new_protected:Nn \__ccool_option_outer:n
261 {
262   \tl_gset:Nn \g__ccool_option_outer_tl {#1}
263 }

```

(End definition for __ccool_option_outer:n.)

```

\__ccool_option_separ:n #1: {< tl1 >}{< tl2 >}{< tl3 >}
264 \tl_new:N \g__ccool_option_separ_tl
265 \cs_new_protected:Nn \__ccool_option_separ:n
266 {
267   \cs_gset:Npn \g__ccool_option_separ_tl {#1}
268 }

```

(End definition for __ccool_option_separ:n.)

```

\g__ccool_option_separ_tl
269 \ifcsdef{text}
270 {
271   \tl_const:Nn \c__ccool_option_separ_default_tl
272   {
273     { \text{{\ } \__ccool_lang_and: {\ } } }
274     { \text{,{\ } } }
275     { \text{,{\ } \__ccool_lang_and: {\ } } }
276   }

```

```

277 }
278 {
279   \tl_const:Nn \c__ccool_option_separ_default_tl
280   {
281     { {\ } \__ccool_lang_and: {\ } }
282     { ,{\ } }
283     { ,{\ } \__ccool_lang_and: {\ } }
284   }
285 }

```

(End definition for `\g__ccool_option_separ_tl`.)

9 prop

```

\__ccool_prop_append:NN #1 : < prop1 >
#2 : < prop2 >

286 \cs_new_protected:Npn \__ccool_prop_append:NN #1 #2
287 {
288   \cs_set:Nn \__ccool_prop_append:nn
289   {
290     \prop_gput:Nnx #1 {##1}{ \prop_item:Nn #2{##1} }
291   }
292   \prop_map_function:NN #2 \__ccool_prop_append:nn
293 }
294 \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 >

295 \cs_new_protected:Nn \__ccool_prop_append:Nn
296 {
297   \__ccool_prop_append:cN{ \__ccool_prop_name:n {#2} } #1
298 }

```

(End definition for `__ccool_prop_append:Nn`.)

```

\__ccool_prop_clear_new:n #1 : < tl var name >

299 \cs_new_protected:Nn \__ccool_prop_clear_new:n
300 {
301   \exp_args:No \prop_clear_new:c{ \__ccool_prop_name:n {#1} }
302 }

```

(End definition for `__ccool_prop_clear_new:n`.)

```

\__ccool_prop_clear_new_map:n #1 : < keyval list >

303 \cs_new_protected:Nn \__ccool_prop_clear_new_map:n
304 {
305   \seq_set_from_clist:Nn \g__ccool_aux_key_seq {#1}
306   \seq_map_function:NN \g__ccool_aux_key_seq \__ccool_prop_clear_new:n
307 }

```

(End definition for `__ccool_prop_clear_new_map:n`.)

```

\__ccool_prop_if_exist:nTF #1 : <tl1>
#2 : <tl2>
#3 : <tl3>

308 \cs_new:Nn \__ccool_prop_if_exist:nTF
309 {
310   \prop_if_exist:cTF{ \__ccool_prop_name:n {#1} }{#2}{#3}
311 }

(End definition for \__ccool_prop_if_exist:nTF.)

```

```

\__ccool_prop_item:nn #1 : <tl var name >
#2 : <key >

312 \cs_new:Nn \__ccool_prop_item:nn
313 {
314   \prop_item:cn { \__ccool_prop_name:n {#1} } {#2}
315 }

(End definition for \__ccool_prop_item:nn.)

```

```

\__ccool_prop_name:n #1 : <tl var name >

316 \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 >

317 \cs_new_protected:Nn \__ccool_prop_new:n
318 {
319   \prop_new:c{ \__ccool_prop_name:n {#1} }
320 }

(End definition for \__ccool_prop_new:n.)

```

10 seq

```

\__ccool_seq_from_prop:NNn #1 : <seq1>
#2 : <seq2> (keys)
#3 : <prop >

321 \cs_new_protected:Nn \__ccool_seq_from_prop:NNn
322 {
323   \cs_set_protected:Nn \__ccool_seq_from_prop:n
324   {
325     \seq_gput_right:No #1 { \prop_item:cn{#3}{##1} }
326   }
327   \seq_map_function:NN #2 \__ccool_seq_from_prop:n
328 }

(End definition for \__ccool_seq_from_prop:NNn.)

```

11 Front-end

`\CcoolClear`

```
329 \NewDocumentCommand{ \CcoolClear }
330 { D<>{\g__ccool_option_param_tl} }
331 {
332   \__ccool_prop_clear_new_map:n{#1}
333 }
```

(End definition for `\CcoolClear`. This function is documented on page 6.)

`\CcoolHook`

```
334 \NewDocumentCommand{\CcoolHook}{}{\c_empty_tl}
```

(End definition for `\CcoolHook`. This function is documented on page 6.)

`\CcoolLambda` (*Note*²)

```
335 \ProvideDocumentCommand \CcoolLambda { 0{m} m }
336 {
337   \erw_lambda:nnn \DeclareDocumentCommand { #1 } { #2 }
338 }
```

(End definition for `\CcoolLambda`. This function is documented on page 6.)

`\CcoolOption` (*Note*³) (*Note*⁴)

```
339 \NewDocumentCommand{ \CcoolOption }
340 { 0{ And, Expans, File, Inner, Param, Outer, Separ, Write } }
341 {
342   \keys_set:nn{ __ccool }{#1}
343 }
```

(End definition for `\CcoolOption`. This function is documented on page 7.)

```
344 \keys_define:nn { __ccool }
345 {
```

And

```
346 And .code:n = { \__ccool_lang_and_update:e{ #1 } },
347 And .default:n = { \c__ccool_lang_and_tl },
348 And .initial:n = { \c__ccool_lang_and_tl },
```

Expans

```
349 Expans .multichoices:nn = { eo, ee, ex, xo, xe, xx }
350 { \tl_gset_eq:NN \g__ccool_option_expans_tl \l_keys_choice_tl },
351 Expans .default:n = { xo },
352 Expans .initial:n = { xo },
```

File

```
353 File .code:n = {
354   \tl_gset:Nx \g__ccool_log_file_tl{#1}
355 },
356 File .default:n = { \erw_sys_jobnametimestamp: },
357 File .initial:n = { \erw_sys_jobnametimestamp: },
```

Inner

```
358 Inner .code:n={
359   \__ccool_option_inner:n{#1}
360   \exp_last_unbraced:Nf
361   \__ccool_make_ccool:nmnn
362   {
363     { \g__ccool_option_param_tl }
364     { \g__ccool_option_inner_tl }
365     { \g__ccool_option_separ_tl }
366     { \g__ccool_option_outer_tl }
367   }
368 },
369 Inner .value_required:n = false,
370 Inner .default:n = {###1},
371 Inner .initial:n = {###1},
```

Param

```
372 Param .code:n={
373   \__ccool_option_param:n{#1}
374   \exp_last_unbraced:Nf
375   \__ccool_make_ccool:nmnn
376   {
377     { \g__ccool_option_param_tl }
378     { \g__ccool_option_inner_tl }
379     { \g__ccool_option_separ_tl }
380     { \g__ccool_option_outer_tl }
381   }
382 },
383 Param .value_required:n = false,
384 Param .default:n = { Default },
385 Param .initial:n = { Default },
```

Outer

```
386 Outer .code:n={
387   \__ccool_option_outer:n{#1}
388   \exp_last_unbraced:Nf
389   \__ccool_make_ccool:nmnn
390   {
391     { \g__ccool_option_param_tl }
392     { \g__ccool_option_inner_tl }
393     { \g__ccool_option_separ_tl }
394     { \g__ccool_option_outer_tl }
395   }
396 },
```

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

```

397 Outer .value_required:n = false,
398 Outer .default:n = { \ensuremath{####1} },
399 Outer .initial:n = { \ensuremath{####1} },

```

Separ

```

400 Separ .code:n={
401   \__ccool_option_separ:n{#1}
402   \exp_last_unbraced:Nf
403   \__ccool_make_ccool:nnnn
404   {
405     { \g__ccool_option_param_tl }
406     { \g__ccool_option_inner_tl }
407     { \g__ccool_option_separ_tl }
408     { \g__ccool_option_outer_tl }
409   }
410 },
411 Separ .value_required:n = false,
412 Separ .default:n = { \c__ccool_option_separ_default_tl },
413 Separ .initial:n = { \c__ccool_option_separ_default_tl },

```

Write

```

414 Write .code:n = {
415   \bool_if:nTF{#1}
416   {\__ccool_log_open:}
417   {\__ccool_log_close:}
418 },
419 Write .value_required:n = false,
420 Write .default:n = \BooleanFalse,
421 Write .initial:n = \BooleanFalse
422 }

```

\CcoolRead

```

423 \NewDocumentCommand{\CcoolRead}
424 {o}
425 {
426   \IfValueTF{#1}
427   {\__ccool_log_read:e{#1}}
428   {\__ccool_log_read:}
429 }

```

(End definition for \CcoolRead. This function is documented on page 8.)

\CcoolVers

```

430 \NewDocumentCommand{\CcoolVers}
431 {}
432 {\use:c{ver@ccool.sty}}

```

(End definition for \CcoolVers. This function is documented on page 8.)

12 Closing

```

433 \ExplSyntaxOff
434 \</package>

```