The keyparse package[*]

Erwann Rogard[†]

Released 2021-08-18

Abstract

This LaTeXpackage provides an interface to define and evaluate key-based replacement rules[1]. It can be used to parse the argument specification of a document command[2].

Contents

[*]This file describes version v1.0, last revised 2021-08-18.

[†]first.lastname at gmail.com

Part I
Usage

1 Document command

`KeyparseKeys` `\KeyparseKeys{⟨rule⟩}`

Expands to The keys associated with ⟨*rule*⟩

`KeyparseEval` `\KeyparseEval`

Adapts `\keyparse_eval:nn`

2 Programming

`\keyparse_set:nnnnn` `\keyparse_set:nnnnn{⟨rule⟩}{⟨key⟩}{⟨signature⟩}{⟨replacement⟩}{⟨recurse⟩}`

Requires ⟨*rule*⟩ is a token list; ⟨*key*⟩⟨*signature*⟩ is a valid “weird” argument specifier[1, Naming functions and variables]; ⟨*replacement*⟩ and ⟨*recurse*⟩ are in terms of ⟨*signature*⟩.

Semantics As shown under **Expands to** for `\keyparse_eval:nn`

Tip Using `{⟨token⟩}` as ⟨*replacement*⟩, one can iterate over the result of `\keyparse_eval:nn` using `\tl_map_function:nN`. Using ⟨*token*⟩, instead, merges the ⟨*tokens*⟩'s.

`\keyparse_eval:nn ⋆` `\keyparse_eval:nn{⟨rule⟩}{⟨key⟩⟨args⟩}`

Requires ⟨*args*⟩ is compatible with ⟨*signature*⟩ for that ⟨*rule*⟩ and ⟨*key*⟩

Expands to ⟨*replacement*⟩`\keyparse_eval:nn{⟨rule⟩}{⟨recurse⟩}`

Other

`\keyparse_argspec_e:n ⋆` `\keyparse_argspec_e:n{⟨token list⟩}`

3 Rule

Hereafter are rules defined with `\keyparse_set:nnnnn`.

`argspec`

Keys `e`, `d`, `m`, `o`, `r`, `s` and `t`

Requires $\langle key_i\rangle\langle arg_i\rangle$ is a valid document-command argument specifier[2]

Rule i $\langle key_i\rangle\langle arg_i\rangle \to$ `{`$\langle key_i\rangle\langle arg_i\rangle$`}`

`pair/first`

Keys `>`

Rule 1 `>{`$\langle$*first*$\rangle$`}{`$\langle$*second*$\rangle$`}` $\to \langle$*first*$\rangle$

`pair/merge`

Keys `>`

Rule 1 `>{`$\langle$*first*$\rangle$`}{`$\langle$*second*$\rangle$`}` $\to \langle$*first*$\rangle\langle$*second*$\rangle$

Part II
Listing

Listing 1. Making rules for &, >{.}, and +.+

```
\ExplSyntaxOn
\group_begin:
\keyparse_set:nnnnn{foo}{&}{#1}{{&}}{#1}
\keyparse_set:nnnnn{foo}{>}{#1#2}{{#1}}{#2}
\keyparse_set:nnnnn{foo}{+}{#1+#2}{{#1}}{#2}
\exp_args:Nx
\tl_map_inline:nn
{\keyparse_eval:nn{foo}{&>{123}+xyz+}}
{\texttt{(\tl_to_str:n{#1})}}
\group_end:
\ExplSyntaxOff
```

```
(&)(123)(xyz)
```

Listing 2. Embedding ‘LaTeX’ in “Leslie Lamport built . . . ”

```
\begingroup
\KeyparseEval{pair/first}
{>{}{Leslie~}
  >{La}{mport~built~LaTeX~on~top~of~Donald~Knuth’s~}
  >{TeX}{.}}
\endgroup
```

LaTeX

Listing 3. argspec

```
\ExplSyntaxOn
\group_begin:
\tl_set:Nx\l_tmpa_tl
{\keyparse_eval:nn{argspec}{msotae{_^}r<]d[>}}
\exp_args:Nx
\tl_map_inline:Nn
\l_tmpa_tl
{\texttt{(\tl_to_str:n{#1})}}
\group_end:
\ExplSyntaxOff
```

```
(m)(s)(o)(ta)(e{_})(e{^})(r<])(d[>)
```

1 Support

This package is available from https://github.com/rogard/keyparse.

Part III
Implementation

```
⟨*package⟩
⟨@@=keyparse⟩
\ExplSyntaxOn
```

1 keyparse

1.1 Interface

not-set

```
\msg_new:nnn
{__keyparse}
{not-set}
{recursion~for~rule~#1~is~not~set}
```

(*End definition for* not-set.)

__keyparse_keyparse_rule_w:n
__keyparse_keyparse_rule:n
__keyparse_keyparse_eval_aux:nn
__keyparse_keyparse_eval_aux:ne

#1 : rule

```
\cs_new_protected:Nn
\__keyparse_keyparse_rule_w:n
{\clist_clear_new:c{__keyparse_keys_#1_clist}
  \cs_new:cpn
  {__keyparse_keyparse_eval_#1:w} ##1 ##2 \q_recursion_stop
  {\quark_if_recursion_tail_stop:n{##1}
    \use:c{__keyparse_keyparse_eval_#1_##1:w}##2\q_recursion_stop }}
\cs_new_protected:Nn
\__keyparse_keyparse_rule:n{\__keyparse_keyparse_rule_w:n{#1}}
\cs_new:Nn
\__keyparse_keyparse_eval_aux:nn
{\cs_if_exist:cTF
  {__keyparse_keyparse_eval_#1:w}
  { \use:c{__keyparse_keyparse_eval_#1:w}#2
    \q_recursion_tail
    \q_recursion_stop}
  {\msg_error:nnn{__keyparse}
    {not-set}
    {#1}}}
\cs_generate_variant:Nn\__keyparse_keyparse_eval_aux:nn{ne}
```

(*End definition for* __keyparse_keyparse_rule_w:n *,* __keyparse_keyparse_rule:n *, and* __keyparse_keyparse_eval_aux:nn*.*)

\keyparse_eval:nn
\keyparse_set:nnnnn

```
\cs_new:Nn
\keyparse_eval:nn
{\__keyparse_keyparse_eval_aux:ne{#1}
  {\tl_trim_spaces:n{#2}}}
\cs_new_protected:Nn
\keyparse_set:nnnnn
{\cs_if_exist:cTF
  {__keyparse_keyparse_eval_#1:w}
  {\clist_put_right:cn
    {__keyparse_keys_#1_clist}{\texttt{\tl_to_str:n{#2}}}
    \cs_new:cpn
    {__keyparse_keyparse_eval_#1_#2:w}#3 \q_recursion_stop
    {#4\use:c{__keyparse_keyparse_eval_#1:w}#5 \q_recursion_stop}}
  {\__keyparse_keyparse_rule:n{#1}
    \keyparse_set:nnnnn
    {#1}{#2}{#3}{#4}{#5}}}
```

(*End definition for* `\keyparse_eval:nn` *and* `\keyparse_set:nnnnn`*. These functions are documented on page 2.*)

\KeyparseKeys
\KeyparseEval

```
\ProvideDocumentCommand
{\KeyparseKeys}
{m}
{\clist_use:cnnn
  {__keyparse_keys_#1_clist}
  {~and~}{,~}{~and~}}
\NewDocumentCommand{\KeyparseEval}
{mm}
{\keyparse_eval:nn{#1}{#2}}
```

(*End definition for* `\KeyparseKeys` *and* `\KeyparseEval`*. These functions are documented on page 2.*)

1.2 argspec

\keyparse_argspec_e:n

```
\cs_new:Nn\keyparse_argspec_e:n{{e{#1}}}
```

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

argspec

Expandability forbids inline, hence `\keyparse_argspec_e:n` for key 'e'

```
\keyparse_set:nnnnn{argspec}{e}{#1#2}
{  \tl_map_function:nN{#1}\keyparse_argspec_e:n}{#2}
\keyparse_set:nnnnn{argspec}{d}{#1#2#3}{{d#1#2}}{#3}
\keyparse_set:nnnnn{argspec}{m}{#1}{{m}}{#1}
\keyparse_set:nnnnn{argspec}{o}{#1}{{o}}{#1}
\keyparse_set:nnnnn{argspec}{r}{#1#2#3}{{r#1#2}}{#3}
\keyparse_set:nnnnn{argspec}{s}{#1}{{s}}{#1}
\keyparse_set:nnnnn{argspec}{t}{#1#2}{{t#1}}{#2}
```

(*End definition for* `argspec`*.*)

1.3 pair

pair/first
pair/merge

```
\keyparse_set:nnnnn{pair/first}{>}{#1#2#3}{#1}{#3}
\keyparse_set:nnnnn{pair/merge}{>}{#1#2#3}{#1#2}{#3}
```

(*End definition for* `pair/first` *and* `pair/merge`*.*)

```
\ExplSyntaxOff
⟨/package⟩
```