The l3keys2e package*
Parsing LaTeX3 keyvals as LaTeX 2ε package options

The LaTeX3 Project†

2009/08/24

1 Key–value arguments as LaTeX 2ε class and package options

The key–value method for optional arguments is very popular, as it allows the class or package author to define a large number of options with a simple interface. The expl3 bundle of LaTeX3 base code includes the module l3keys for defining keys, but to use these when loading LaTeX 2ε packages and classes requires extra support. That support is provided by this small package, which is intended to enable LaTeX 2ε packages to benefit from the power of the LaTeX3 key–value system.

1.1 Creating and using keyval options

As with any key–value input, using key–value pairs as package or class options has two parts. The first stage is to define one or more keys. This should be done using the LaTeX3 module l3keys. For example, an option which simply stores a value would be created using:

```
\keys_define:nn { module } {
  option .set:N = \l_module_variable_tl
}
```

On its own, this will not make the key an option for the package or class containing the definition. The second stage is therefore to process the current options, searching for applicable keys.

*This file has version number 1494, last revised 2009/08/24.

†Frank Mittelbach, Denys Duchier, Chris Rowley, Rainer Schöpf, Johannes Braams, Michael Downes, David Carlisle, Alan Jeffrey, Morten Høgholm, Thomas Lotze, Javier Bezos, Will Robertson, Joseph Wright

`\ProcessKeysOptions` `\ProcessKeysOptions {`⟨*module*⟩`}`

The `\ProcessKeysOptions` function is used to check the current option list against the keys defined for ⟨*module*⟩. Global (class) options and local (package) options are checked when this function is called in a package. Each option which does match a key name is then used to attempt to set the appropriate key using `\keys_set:nn`. For example, the option defined earlier would be processed by the line

```
\ProcessKeysOptions { module }
```

1.2 Internal functions and variables

`\keys_latexe_options_clist` Local options and applicable global (class) options are collected up here prior to processing using `\keys_set:nn`.

`\keys_latexe_options:n` `\keys_latexe_options:n {`⟨*module*⟩`}`

Examines current global and local options, comparing them to the keys defined for ⟨*module*⟩. Option names which match defined keys are then executed by passing the appropriate data to `\keys_set:nn`.

`\keys_latexe_options_global:n` `\keys_latexe_options_global:n {`⟨*module*⟩`}`

Makes appropriate checks on global (class) options, then searches for key matches for ⟨*module*⟩.

`\keys_latexe_options_class:n` `\keys_latexe_options_class:n {`⟨*module*⟩`}`

Processes global options when the current file is itself a class, checking the options against keys defined for ⟨*module*⟩.

`\keys_latexe_options_package:n` `\keys_latexe_options_package:n {`⟨*module*⟩`}`

Processes global options when the current file is a LaTeX 2$_\varepsilon$ package, checking the options against keys defined for ⟨*module*⟩.

`\keys_latexe_options_local:` `\keys_latexe_options_local:`

Finds local (package) options and adds them to the list to be parsed by `\keys_set:nn`.

\keys_latexe_remove_equals:n ⋆
\keys_latexe_remove_equals:w ⋆

\keys_latexe_remove_equals:n {⟨*option*⟩}
\keys_latexe_remove_equals:w ⟨*key*⟩ = ⟨*value*⟩ \q_stop

Returns only the key part of a key–value option by removing everything after the first equals sign. It is assumed that, as this will apply in the preamble, equals signs have category code 12.

1.3 Implementation

The usual lead-off, using xparse at the interface level.

```
⟨*package⟩
\ProvidesExplPackage
  {\filename}{\filedate}{\fileversion}{\filedescription}
\RequirePackage{xparse}
```

\keys_latexe_options_clist

The module collects all applicable global and local options into a single list before processing them.

```
\clist_new:N \keys_latexe_options_clist
```

\keys_latexe_options:n

The main function calls functions to collect up the global and local options into \keys_latexe_options_clist before calling the main l3keys module to actually do the processing. So that a suitable message is produced if the option is unknown, the special unknown key is set if it does not already exist for the current module.

```
\cs_new:Npn \keys_latexe_options:n #1 {
  \clist_clear:N \keys_latexe_options_clist
  \keys_latexe_options_global:n {#1}
  \keys_latexe_options_local:
  \keys_if_exist:nnF {#1} { unknown } {
    \keys_define:nn {#1} {
      unknown .code:n = {
        \msg_warning:nnxx { Option~Processing } { unknown-option }
          { \l_keys_key_tl } { \@currname }
      }
    }
  }
  \keys_set:nV {#1} \keys_latexe_options_clist
  \AtEndOfPackage { \cs_set_eq:NN \@unprocessedoptions \scan_stop: }
}
\msg_new:nnn { Option~Processing } { unknown-option } {%
  Unknown~option~‘#1’\\%
  for~module~#2.%
}
```

`\keys_latexe_options_global:n` Global (class) options are handled differently for LaTeX 2_ε packages and classes. Hence this function is essentially a check on the current file type. The initial test is needed as LaTeX 2_ε allows variables to be equal to `\scan_stop:`, which is forbidden in LaTeX3 code.

```
\cs_new:Npn \keys_latexe_options_global:n #1 {
  \cs_if_eq:NNF \@classoptionslist \scan_stop: {
    \cs_if_eq:NNTF \@currext \@clsextension {
      \keys_latexe_options_class:n {#1}
    }{
      \keys_latexe_options_package:n {#1}
    }
  }
}
```

`\keys_latexe_options_class:n` For classes, each option (stripped of any content after '=') is checked for existence as a key. If found, the option is added to the combined list for processing. On the other hand, unused options are stored up in `\@unusedoptionlist`.

```
\cs_new:Npn \keys_latexe_options_class:n #1 {
  \clist_map_inline:Nn \@classoptionslist {
    \keys_if_exist:nnTF {#1} { \keys_latexe_remove_equals:n {##1} } {
      \clist_put_right:Nn \keys_latexe_options_clist {##1}
    }{
      \clist_put_right:Nn \@unusedoptionlist {##1}
    }
  }
}
```

`\keys_latexe_options_package:n` For global options when processing a package, the tasks are slightly different from those for a class. The check is the same, but here there is nothing to do if the option is not applicable. Each valid option also needs to be removed from `\@unusedoptionlist`.

```
\cs_new:Npn \keys_latexe_options_package:n #1 {
  \clist_map_inline:Nn \@classoptionslist {
    \keys_if_exist:nnT {#1} { \keys_latexe_remove_equals:n {##1} } {
      \clist_put_right:Nn \keys_latexe_options_clist {##1}
      \clist_remove_element:Nn \@unusedoptionlist {##1}
    }
  }
}
```

`\keys_latexe_options_local:` If local options are found, the are added to the processing list. LaTeX 2_ε stores options for each file in a macro which may or may not exist, hence the need to use `\cs_if_exist:c`.

```
\cs_new_nopar:Npn \keys_latexe_options_local: {
  \cs_if_eq:NNF \@currext \@clsextension {
    \cs_if_exist:cT { opt@ \@currname . \@currext } {
      \exp_args:NNc \clist_put_right:NV \keys_latexe_options_clist
        { opt@ \@currname . \@currext }
```

```
    }
  }
}
```

`\keys_latexe_remove_equals:n`
`\keys_latexe_remove_equals:w`

As the name suggests, this is a simple function to remove an equals sign from the input. This is all wrapped up in an `n` function so that there will always be a sign available.

```
\cs_new:Npn \keys_latexe_remove_equals:n #1 {
  \keys_latexe_remove_equals:w #1 = \q_stop
}
\cs_new:Npn \keys_latexe_remove_equals:w #1 = #2 \q_stop {#1}
```

`\ProcessKeysOptions`

The user macro is simply a wrapper around the internal process. In contrast to other similar packages, the module name is always required here.

```
\NewDocumentCommand \ProcessKeysOptions { m } {
  \keys_latexe_options:n {#1}
}
\@onlypreamble \ProcessKeysOptions
```

⟨/package⟩

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.