The keyval2e Package☆

Robust and fast key parser

Ahmed Musa✉
Preston, Lancashire, UK

24th August 2011

Contents

1 Introduction

The keyval2e package provides lightweight and robust facilities for creating and managing keys. Its machinery isn't as extensive as that of, e. g., ltxkeys package but it is equally robust. Ease of use and speed of processing are the two main motives of this package. Some, indeed many, applications of the key-value syntax (while they call for robustness) don't require the full armor of key-value processing as found in, e. g., the ltxkeys package. This package was prompted by a subscriber's post on comp.text.tex in August 2011.

In the keyval2e package, command, boolean, and choice keys can be created using only one command (\kve@definekeys). Keys can be initialized with their default values (with the command \kve@setdefaults) as soon as they are created, or at any time. And in any run the default values of keys can be used to set keys that have no current values. The latter task is accomplished by the command \kve@setafterdefaults, meaning 'set keys with their current user-supplied values after the absent keys (i. e., those without current values) have been initialized/set with their default values.'

Keys can be set with the re-entrant command \kve@setkeys, but this will not automatically set up the absent keys (i. e., keys not submitted to the command in its current run) with their

☆ The package is available at http://www.ctan.org/tex-archive/macros/latex/contrib/keyval2e/. This user manual corresponds to version 0.0.2.

default values. To set keys up with their default values, the user has to call \kve@setdefaults or \kve@setafterdefaults.

The keyval2e package has no provision for processing package or class options. See the ltxkeys package for this service. I have seen users who require the services of keys only in documents, and not in package or class files. And some package authors still use LaTeX's native option processing schemes. For those authors, the keyval2e package may be used to process keys (but not options) in package and class files. Since the catoptions package is loaded by the keyval2e package and the former has a robust and extensive options parsing scheme, it may be used for the options processing requirements of the user.

The keyval2e package provides handy tools for creating commands based on the infrastructure of keys. See the file keyval2e-examples.tex for examples. The so-called 'key commands' (see keycommand and skeycommand packages) can be created rather easily with the facilities of this package.

The keyval package provides a simple and widely used interface, but it is not robust, in the sense that it strips off outer curly braces in key values. Also, it has no means to automatically call up default key values after the keys have been defined. Moreover, it automatically redefines existing keys.

2 Package options

The keyval2e package currently has no options.

3 User commands

As previously mentioned in section 1, the keyval2e package can be used to directly define only command and boolean keys. Choice keys can, however, be created indirectly as command keys by using the \kve@checkchoice command (see subsection 3.1). For the user, the only difference between command and ordinary keys is that command keys define macros to hold the user input, making command keys more attractive than ordinary keys.

The only key-defining command in this package is \kve@definekeys. This command distinguishes a boolean key from command keys by the default value of the boolean key. Therefore, all boolean keys must have default values in the set {true | false}, otherwise they will be treated as command keys. Command keys may have no default values and no callbacks.

New macros: \kve@definekeys, \kve@setkeys, etc

```
\kve@definekeys[⟨pref⟩]{⟨fam⟩}[⟨mp⟩]{%
  ⟨key-1⟩/⟨dft-1⟩/⟨cbk-1⟩,
  ...,
  ⟨key-n⟩/⟨dft-n⟩/⟨cbk-n⟩
}

\kve@definekeys*[⟨pref⟩]{⟨fam⟩}[⟨mp⟩]{%
  ⟨key-1⟩/⟨dft-1⟩/⟨cbk-1⟩,
  ...,
  ⟨key-n⟩/⟨dft-n⟩/⟨cbk-n⟩
}

\kve@setkeys[⟨pref⟩]{⟨fam⟩}[⟨na⟩]{⟨keyval⟩}

\kve@setdefaults[⟨pref⟩]{⟨fam⟩}[⟨na⟩]
```

13 `\kve@setafterdefaults[⟨pref⟩]{⟨fam⟩}[⟨na⟩]{⟨keyval⟩}`

Here, ⟨pref⟩ is the optional key prefix (its default is KV), ⟨fam⟩ is the mandatory family, ⟨mp⟩ is the key-value-holding macro prefix (its default is kvmp@), ⟨dft-i⟩ is the default value of key 'i', ⟨cbk-i⟩ is the callback (i. e., the function that will be executed when the key is set) of key 'i', and ⟨keyal⟩ is a list of ⟨key⟩=⟨value⟩ pairs.

⟨na⟩ is a comma-separated list of keys that should be ignored, ie, not set in the current run of setting keys*. \⟨mp⟩@⟨key⟩ will hold the current value of ⟨key⟩. The key macro (i. e., the macro that holds the key's callback) is always \⟨pref⟩@⟨fam⟩@⟨key⟩.

The starred (⋆) variant of the command \kve@definekeys will define only definable keys, in the sense of LaTeX's \newcommand. In that case the commands \⟨mp⟩@⟨key⟩ must also be unique, i. e., not previously defined. The plain form \kve@definekeys will always define the key, whether or not the key already exists; existing keys will thus be overwritten in this case.

You can use '#1' in ⟨cbk⟩ to access the user-supplied value of the current key. Also the macros \currpref, \currfam, \currkey, \currval and \currkeyval are always available when setting keys and may be called in ⟨cbk⟩ at key definition time.

Note 3.1 The list parser for the command \kve@definekeys is comma ','. Hence, if you have literal comma in ⟨cbk⟩, the ⟨cbk⟩ has to be enclosed in curly braces. Leading and trailing spaces in the elements are removed in the internal processing. Explicit spaces (i. e., those needed by the key user) will therefore need to be wrapped in curly braces.

The command \kve@setdefaults will set all the keys in the given family ⟨fam⟩ and prefix ⟨pref⟩ with their default values. All boolean keys (i. e., those with a default in the set {true | false}) will be initialized with a default value of false. This is to avoid premature toggling of the state of such keys. The command \kve@setwithdefaults is an alias for \kve@setdefaults.

Note 3.2 After the keys have been defined, they are automatically set with their default values using the command \kve@setdefaults. This provides default definitions for immediate use.

The command \kve@setafterdefaults will set the given ⟨key⟩=⟨value⟩ pairs after initializing to default values all those keys (in the given family and prefix) that are not listed in the accompanying ⟨key⟩=⟨value⟩ list. This provides a mechanism for (re)initialing to default values those keys that don't have values in ⟨key⟩=⟨value⟩. This type of (re)initialization is often required in the deployment of keys—since the immediate past user values of the keys may no longer be valid. It is useful to have a handy way of accomplishing this task semi-automatically.

3.1 Utility macros

The following macros are utilities.

New macro: \kve@checkchoice

14 `\kve@checkchoice{⟨teststring⟩}{⟨nominations⟩}{⟨nomatch⟩}`

The expandable command \kve@checkchoice can be used to create choice keys as command keys. The ⟨nominations⟩ have the syntax

*When setting keys, undefined keys are reported by the keyval2e package as undefined and are not saved as 'remaining keys', unlike in the ltxkeys package. Moreover, there are no 'undefined key handlers' and no 'handled keys' in this package. Please see the ltxkeys package for these features.

Nominations and callbacks

```
15 ⟨nom-1⟩:⟨cbk-1⟩,...,⟨nom-n⟩:⟨cbk-n⟩
```

Here, please note the colon ':', which separates ⟨nom⟩ from ⟨cbk⟩. ⟨cbk-i⟩ will be executed if ⟨nom-i⟩ matches ⟨teststring⟩. The first match found takes priority over subsequent matches. The fallback ⟨nomatch⟩ will be executed if ⟨teststring⟩ doesn't match any of the ⟨nom⟩'s.

New macro: \kve@checkbool

```
16 \kve@checkbool{⟨val⟩}{⟨true⟩}{⟨false⟩}
```

This checks if ⟨val⟩ is an admissible value of a boolean, namely, if it is in the set {true | false}. If ⟨val⟩ is valid, the text ⟨true⟩ will be executed; otherwise ⟨false⟩ will be executed.

Note 3.3 The user-supplied values of all boolean keys are automatically checked by this command. Hence, the user doesn't have to call this command repeatedly to confirm the validity of values of boolean keys.

New macro: \kve@keyvalerr

```
17 \kve@keyvalerr
```

This is a parameterless command that uses \currkey and \currval internally. It simply generates an error to indicate that the current value of a key is invalid. It will indicate the key name and the truncated version of the key value that is invalid.

4 Examples

Example: \kve@definekeys

```
18 \kve@definekeys[KV]{fam}[mp@]{%
19   % keya and keyb are boolean keys. They will call \kve@checkbool
20   % internally to check the user input for them. keya has no callback:
21   keya/true,
22   keyb/false/\ifmp@keyb\def\x{found}\else\def\x{not found}\fi,
23   % keyc is a choice key:
24   keyc/left/\kve@checkchoice{#1}{left:\let\x\flushleft,
25     right:\let\x\flushright}{\kve@keyvalerr},
26   % keyd has no default. Therefore, it can't be set without a user value.
27   % In \kve@setdefaults we set it with a default value of 'empty', but
28   % its user must always provide a value for it:
29   keyd,
30   % keyone has an empty default value. This doesn't mean 'no default':
31   keyone//\ifnullTF{#1}{\def\x{empty}}{\def\x{#1}},
32   % keytwo has no callback:
33   keytwo/+,
34   % keythree has a braced default value:
35   keythree/{left}/\def\y##1{'para-scientific gobbledegook' ##1},
36   % keyfour sets keyone (see note 4.1):
37   keyfour/left/\kve@setkeys[KV]{fam}{keyone=#1},
38 }
```

Remember that after the keys have been defined, they are automatically set with their default values using the command \kve@setdefaults.

Note 4.1 The type of re-entrance staged by key keyfour above should in general be done with care, otherwise you could end up with infinite re-entrance. Therefore, the package sets a re-entrance limit of 4, to alert the user to the probability that an infinite loop has been created by him in using \kve@setkeys. In the unlikely event that you need to exceed this limit, then please turn to the ltxkeys package.

The following command says 'set the given keys with their current values, after the absent keys of the given family and prefix have been set up with their default values'. Keys with current values will not be set with their default values:

Example: \kve@setafterdefaults

```
39 \kve@setafterdefaults[KV]{fam}{keyone=+,keytwo=abc,keythree=+,keyfour=xax}
```

Please see keyval2e-examples.tex for the fuller version of the following example:

Examples: Creating a key command

```
40 \documentclass{minimal}
41 \usepackage{keyval2e}
42 \makeatletter
43 \kve@definekeys*[KV]{fam}[mp@]{%
44   keyone/+,
45   keytwo/+,
46   keythree/+,
47   keyfour/+
48 }
49 \def\fourplus{+,+,+,+}
50 \newcommand{\test}[2]{%
51   \kve@setafterdefaults[KV]{fam}{#2}%
52   \edef\tempa{\mp@keyone,\mp@keytwo,\mp@keythree,\mp@keyfour}%
53   Test #1: *\texttt{\tempa}*%
54   \ifxTF\tempa\fourplus{All values are defaults}{At least one value is set}%
55 }
56 \begin{document}
57 \ttfamily\noindent
58 \test{A}{}\\
59 \test{B}{keythree=+}\\
60 \test{C}{keythree=a}\\
61 \end{document}
```

5 Version history

The star sign (*) on the right-hand side of the following lists means the subject features in the package but is not reflected anywhere in this user guide.

Version 0.0.2 [2011/08/22]

Automatically call \kve@checkbool when setting boolean keys . . . . . subsection 3.1

Raise error for keys that have no user input and no default value . . . . . section 4

Version 0.0.1a [2011/08/14]

Completed the user guide . . . . . ⋆

Version 0.0.1 [2011/08/13]

First public release . . . . . ⋆

Index

Index numbers refer to page numbers.