The **kvsetkeys** package

Heiko Oberdiek
<heiko.oberdiek at googlemail.com>

2010/01/28 v1.8

Abstract

Package kvsetkeys provides \kvsetkeys, a variant of package keyval's \setkeys. It allows to specify a handler that deals with unknown options. Active commas and equal signs may be used (e.g. see babel's shorthands) and only one level of curly braces is removed from the values.

Contents

1 Documentation

First I want to recommend the very good review article "A guide to key-value methods" by Joseph Wright [1]. It introduces the different key-value packages and compares them.

1.1 Motivation

\kvsetkeys serves as replacement for keyval's \setkeys. It basically uses the same syntax. But the implementation is more robust and predictable:

Active syntax characters: Comma ',' and the equals sign '=' are used inside key value lists as syntax characters. Package keyval uses the catcode of the characters that is active during package loading, usually this is catcode 12 (other). But it can happen that the catcode setting of the syntax characters changes. Especially active characters are of interest, because some language adaptations uses them. For example, option turkish of package babel uses the equals sign as active shorthand character. Therefore package kvsetkeys deals with both catcode settings 12 (other) and 13 (active).

Brace removal: Package keyval's \setkeys removes up to two levels of curly braces around the value in some unpredictable way:

```
\setkeys{fam}{key={{value}}}    → value
\setkeys{fam}{key={{{value}}}}  → {value}
\setkeys{fam}{key= {{{value}}}} → {{value}}
```

This package kvsetkeys follows a much stronger rule: Exactly one level of braces are removed from an item, if the item is surrounded by curly braces. An item can be a the key value pair, the key or the value.

```
\kvsetkeys{fam}{key={value}}   → value
\kvsetkeys{fam}{key={{value}}  → {value}
\kvsetkeys{fam}{key= {{value}} → {value}
```

Arbitrary values: Unmatched conditionals are supported.

Before I describe \kvsetkeys in more detail, first I want to explain, how this package deals with key value lists. For the package also provides low level interfaces that can be used by package authors.

1.2 Normalizing key value lists

`\kv@normalize {⟨key value list⟩}`

If the user specifies key value lists, he usually prefers nice formatted source code, e.g.:

```
\hypersetup{
  pdftitle    = {...},
  pdfsubject  = {...},
  pdfauthor   = {...},
  pdfkeywords = {...},
  ...
}
```

Thus there can be spaces around keys, around `=` or around the value. Also empty entries are possible by too many commas. Therefore these spaces and empty entries are silently removed by package keyval and this package. Whereas the contents of the value can be protected by curly braces, especially if spaces or commas are used inside, a key name must not use spaces or other syntax characters.

`\kv@normalize` takes a key value list and performs the cleanup:

- Spaces are removed.

- Syntax characters (comma and equal sign) that are active are replaced by the same characters with standard catcode. (Example: babel's language option turkish uses the equal sign as active shorthand character.)

The result is stored in `\kv@list`, e.g.:

`\kv@list → ,pdftitle={...},pdfsubject={...},...,`

Curly braces around values (or keys) remain untouched.

v1.3+: One comma is added in front of the list and each pair ends with a comma. Thus an empty list consists of one comma, otherwise two commas encloses the list. Empty entries other than the first are removed.

v1.0 − v1.2: Empty entries are removed later. In fact it adds a comma at the begin and end to protect the last value and an easier implementation.

1.3 Parsing key value lists

`\kv@parse {⟨key value list⟩} {⟨processor⟩}`

It is easier to parse a normalized list, thus `\kv@parse` normalizes the list and calls `\kv@parse@normalized`.

`\kv@parse@normalized {⟨key value list⟩} {⟨processor⟩}`

Now the key value list is split into single key value pairs. For further processing the key and value are given as arguments for the ⟨processor⟩:

⟨processor⟩ {⟨key⟩} {⟨value⟩}

Also key and value are stored in macro names:

- `\kv@key` stores the key.

- `\kv@value` stores the value or if the value was not specified it has the meaning `\relax`.

The behaviour in pseudo code:

```
foreach (⟨key⟩, ⟨value⟩) in (⟨key value list⟩)
    \kv@key := ⟨key⟩
    \kv@value := ⟨value⟩
    ⟨processor⟩ {⟨key⟩} {⟨value⟩}
```

1.4 Processing key value pairs

| \kv@processor@default {⟨family⟩} {⟨key⟩} {⟨value⟩} |

There are many possiblities to process key value pairs. `\kv@processor@default` is the processor used in `\kvsetkeys`. It reimplements and extends the behaviour of keyval's `\setkeys`. In case of unknown keys `\setkeys` raise an error. This processer, however, calls a handler instead, if it is provided by the family.

The behaviour in pseudo code:

```
if ⟨key⟩ exists
    call the keyval code of ⟨key⟩
else
    if ⟨handler⟩ for ⟨family⟩ exists
        ⟨handler⟩ {⟨key⟩} {⟨value⟩}
    else
        raise unknown key error
    fi
fi
```

1.5 Default family handler

`\kv@processor@default` calls ⟨handler⟩, the default handler for the family, if the key does not exist in the family. The handler is called with two arguments, the key and the value. It can be defined with `\kv@set@family@hander`:

| \kv@set@family@handler {⟨family⟩} {⟨handler definition⟩} |

This sets the default family handler for the keyval family ⟨family⟩. Inside ⟨handler definition⟩ `#1` stands for the key and `#2` is the value. Also `\kv@key` and `\kv@value` can be used for the key and the value. If the value is not given, `\kv@value` has the meaning `\relax`.

1.6 Put it all together

| \kvsetkeys {⟨family⟩} {⟨key value list⟩} |

The work is done by the previous commands. `\kvsetkeys` just calls them:

```
\kv@parse {⟨key value list⟩}{\kv@processor@default {⟨family⟩}}
```

Thus you can replace `\setkeys` of package keyval by the key value parser of this package:

```
\renewcommand*{\setkeys}{\kvsetkeys}
or
\let\setkeys\kvsetkeys
```

1.7 Comma separated lists

Since version 2007/09/29 v1.3 this package also supports the normalizing and parsing of general comma separated lists.

`\comma@normalize {⟨comma list⟩}`

Macro `\comma@normalize` normalizes the comma separated list, removes spaces around commas. The result is put in macro `\comma@list`.

`\comma@parse {⟨comma list⟩} {⟨processor⟩}`

Macro `\comma@parse` first normalizes the comma separated list and then parses the list by calling `\comma@parse@normalized`.

`\comma@parse@normalized {⟨normalized comma list⟩} {⟨processor⟩}`

The list is parsed. Empty entries are ignored. ⟨processor⟩ is called for each non-empty entry with the entry as argument:

⟨processor⟩{⟨entry⟩}

Also the entry is stored in the macro `\comma@entry`.

2 Example

The following example prints a short piece of HTML code using the tabbing environment for indenting purpose and a key value syntax for specifying the attributes of an HTML tag. The example illustrates the use of a default family handler.

```
 1 ⟨*example⟩
 2 \documentclass{article}
 3 \usepackage[T1]{fontenc}
 4 \usepackage{kvsetkeys}
 5 \usepackage{keyval}
 6
 7 \makeatletter
 8 \newcommand*{\tag}[2][]{%
 9   % #1: attributes
10   % #2: tag name
11   \begingroup
12     \toks@={}%
13     \let\@endslash\@empty
14     \kvsetkeys{tag}{#1}%
15     \texttt{%
16       \textless #2\the\toks@\@endslash\textgreater
17     }%
18   \endgroup
19 }
20 \kv@set@family@handler{tag}{%
21   % #1: key
22   % #2: value
23   \toks@\expandafter{%
24     \the\toks@
25     \space
26     #1=\string"#2\string"%
27   }%
28 }
29 \define@key{tag}{/}[]{%
30   \def\@endslash{/}%
```

```
31 }
32 \makeatother
33
34 \begin{document}
35 \begin{tabbing}
36   \mbox{}\qquad\=\qquad\=\kill
37   \tag{html}\\
38   \>\dots\\
39   \>\tag[border=1]{table}\\
40   \>\>\tag[width=200, span=3, /]{colgroup}\\
41   \>\>\dots\\
42   \>\tag{/table}\\
43   \>\dots\\
44   \tag{/html}\\
45 \end{tabbing}
46 \end{document}
47 ⟨/example⟩
```

3 Implementation

3.1 Identification

```
48 ⟨*package⟩
```

Reload check, especially if the package is not used with LaTeX.

```
49 \begingroup
50   \catcode44 12 % ,
51   \catcode45 12 % -
52   \catcode46 12 % .
53   \catcode58 12 % :
54   \catcode64 11 % @
55   \catcode123 1 % {
56   \catcode125 2 % }
57   \expandafter\let\expandafter\x\csname ver@kvsetkeys.sty\endcsname
58   \ifx\x\relax % plain-TeX, first loading
59   \else
60     \def\empty{}%
61     \ifx\x\empty % LaTeX, first loading,
62       % variable is initialized, but \ProvidesPackage not yet seen
63     \else
64       \catcode35 6 % #
65       \expandafter\ifx\csname PackageInfo\endcsname\relax
66         \def\x#1#2{%
67           \immediate\write-1{Package #1 Info: #2.}%
68         }%
69       \else
70         \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
71       \fi
72       \x{kvsetkeys}{The package is already loaded}%
73       \aftergroup\endinput
74     \fi
75   \fi
76 \endgroup
```

Package identification:

```
77 \begingroup
78   \catcode35 6 % #
79   \catcode40 12 % (
80   \catcode41 12 % )
81   \catcode44 12 % ,
82   \catcode45 12 % -
83   \catcode46 12 % .
84   \catcode47 12 % /
```

```
85  \catcode58 12 % :
86  \catcode64 11 % @
87  \catcode91 12 % [
88  \catcode93 12 % ]
89  \catcode123 1 % {
90  \catcode125 2 % }
91  \expandafter\ifx\csname ProvidesPackage\endcsname\relax
92    \def\x#1#2#3[#4]{\endgroup
93      \immediate\write-1{Package: #3 #4}%
94      \xdef#1{#4}%
95    }%
96  \else
97    \def\x#1#2[#3]{\endgroup
98      #2[{#3}]%
99      \ifx#1\@undefined
100       \xdef#1{#3}%
101     \fi
102     \ifx#1\relax
103       \xdef#1{#3}%
104     \fi
105   }%
106 \fi
107 \expandafter\x\csname ver@kvsetkeys.sty\endcsname
108 \ProvidesPackage{kvsetkeys}%
109   [2010/01/28 v1.8 Key value parser (HO)]
110 \begingroup
111   \catcode123 1 % {
112   \catcode125 2 % }
113   \def\x{\endgroup
114     \expandafter\edef\csname KVS@AtEnd\endcsname{%
115       \catcode35 \the\catcode35\relax
116       \catcode64 \the\catcode64\relax
117       \catcode123 \the\catcode123\relax
118       \catcode125 \the\catcode125\relax
119     }%
120   }%
121 \x
122 \catcode35 6 % #
123 \catcode64 11 % @
124 \catcode123 1 % {
125 \catcode125 2 % }
126 \def\TMP@EnsureCode#1#2{%
127   \edef\KVS@AtEnd{%
128     \KVS@AtEnd
129     \catcode#1 \the\catcode#1\relax
130   }%
131   \catcode#1 #2\relax
132 }
133 \TMP@EnsureCode{36}{3}% $
134 \TMP@EnsureCode{38}{4}% &
135 \TMP@EnsureCode{39}{12}% '
136 \TMP@EnsureCode{44}{12}% ,
137 \TMP@EnsureCode{46}{12}% .
138 \TMP@EnsureCode{47}{12}% /
139 \TMP@EnsureCode{61}{12}% =
140 \TMP@EnsureCode{94}{7}% ^ (superscript)
141 \TMP@EnsureCode{96}{12}% `
142 \TMP@EnsureCode{126}{13}% ~ (active)
```

3.2 Package loading

```
143 \begingroup\expandafter\expandafter\expandafter\endgroup
144 \expandafter\ifx\csname RequirePackage\endcsname\relax
```

```
145   \input infwarerr.sty\relax
146   \input etexcmds.sty\relax
147 \else
148   \RequirePackage{infwarerr}[2007/09/09]%
149   \RequirePackage{etexcmds}[2010/01/28]%
150 \fi

151 \expandafter\ifx\csname toks@\endcsname\relax
152   \toksdef\toks@=0 %
153 \fi
```

3.3   Check for ε-TEX

\unexpanded, \ifcsname, and \unless are used if found.

```
154 \begingroup\expandafter\endgroup
155 \ifcase0\ifetex@unexpanded
156            \expandafter\ifx\csname ifcsname\endcsname\relax
157          \else
158            \expandafter\ifx\csname unless\endcsname\relax
159            \else
160              1%
161            \fi
162          \fi
163        \fi
164   \catcode`\$=9 % ignore
165   \catcode`\&=14 % comment
166 \else % e-TeX
167   \catcode`\$=14 % comment
168   \catcode`\&=9  % ignore
169 \fi
```

3.4   Generic help macros

\KVS@Empty

```
170 \def\KVS@Empty{}
```

\KVS@FirstOfTwo

```
171 \long\def\KVS@FirstOfTwo#1#2{#1}
```

\KVS@SecondOfTwo

```
172 \long\def\KVS@SecondOfTwo#1#2{#2}
```

\KVS@IfEmpty

```
173 \def\KVS@IfEmpty#1{%
174 & \edef\KVS@Temp{\etex@unexpanded{#1}}%
175 $ \begingroup
176 $   \toks@{#1}%
177 $   \edef\KVS@Temp{\the\toks@}%
178 $ \expandafter\endgroup
179   \ifx\KVS@Temp\KVS@Empty
180     \expandafter\KVS@FirstOfTwo
181   \else
182     \expandafter\KVS@SecondOfTwo
183   \fi
184 }
```

3.5   Normalizing

\kv@normalize

```
185 \def\kv@normalize#1{%
186   \begingroup
187     \toks@{,#1,}%
```

```
188     \KVS@Comma
189     \KVS@SpaceComma{ }%
190     \KVS@CommaSpace
191     \KVS@CommaComma
192     \KVS@Equals
193     \KVS@SpaceEquals{ }%
194     \KVS@EqualsSpace
195     \xdef\KVS@Global{\the\toks@}%
196   \endgroup
197   \let\kv@list\KVS@Global
198 }
```

\comma@normalize

```
199 \def\comma@normalize#1{%
200   \begingroup
201     \toks@{,#1,}%
202     \KVS@Comma
203     \KVS@SpaceComma{ }%
204     \KVS@CommaSpace
205     \KVS@CommaComma
206     \xdef\KVS@Global{\the\toks@}%
207   \endgroup
208   \let\comma@list\KVS@Global
209 }
```

\KVS@Comma Converts active commas into comma with catcode other. Also adds a comma at the end to protect the last value for next cleanup steps.

```
210 \begingroup
211   \lccode'\,='\,%
212   \lccode'\~='\,%
213 \lowercase{\endgroup
214   \def\KVS@Comma{%
215     \toks@\expandafter{\expandafter}\expandafter
216     \KVS@@Comma\the\toks@~\KVS@Nil
217   }%
218   \def\KVS@@Comma#1~#2\KVS@Nil{%
219     \toks@\expandafter{\the\toks@#1}%
220     \KVS@IfEmpty{#2}{%
221     }{%
222       \KVS@@Comma,#2\KVS@Nil
223     }%
224   }%
225 }
```

\KVS@SpaceComma Removes spaces before the comma, may add commas at the end.

```
226 \def\KVS@SpaceComma#1{%
227   \expandafter\KVS@@SpaceComma\the\toks@#1,\KVS@Nil
228 }
```

\KVS@@SpaceComma

```
229 \def\KVS@@SpaceComma#1 ,#2\KVS@Nil{%
230   \KVS@IfEmpty{#2}{%
231     \toks@{#1}%
232   }{%
233     \toks@{#1,#2}%
234     \expandafter\KVS@@SpaceComma\the\toks@\KVS@Nil
235   }%
236 }
```

\KVS@CommaSpace Removes spaces after the comma, may add commas at the end.

```
237 \def\KVS@CommaSpace{%
```

```
238    \expandafter\KVS@@CommaSpace\the\toks@, \KVS@Nil
239 }
```

\KVS@@CommaSpace

```
240 \def\KVS@@CommaSpace#1, #2\KVS@Nil{%
241    \KVS@IfEmpty{#2}{%
242       \toks@{#1}%
243    }{%
244       \toks@{#1,#2}%
245       \expandafter\KVS@@CommaSpace\the\toks@\KVS@Nil
246    }%
247 }
```

\KVS@CommaComma   Replaces multiple commas by one comma.

```
248 \def\KVS@CommaComma{%
249    \expandafter\KVS@@CommaComma\the\toks@,\KVS@Nil
250 }
```

\KVS@@CommaComma

```
251 \def\KVS@@CommaComma#1,,#2\KVS@Nil{%
252    \toks@{#1,#2}%
253    \KVS@IfEmpty{#2}{%
254    }{%
255       \expandafter\KVS@@CommaComma\the\toks@\KVS@Nil
256    }%
257 }
```

\KVS@Equals   Converts active equals signs into catcode other characters.

```
258 \begingroup
259    \lccode`\==`\=%
260    \lccode`\~=`\=%
261 \lowercase{\endgroup
262    \def\KVS@Equals{%
263       \toks@\expandafter{\expandafter}\expandafter
264       \KVS@@Equals\the\toks@~\KVS@Nil
265    }%
266    \def\KVS@@Equals#1~#2\KVS@Nil{%
267       \edef\KVS@Temp{\the\toks@}%
268       \ifx\KVS@Temp\KVS@Empty
269          \expandafter\KVS@FirstOfTwo
270       \else
271          \expandafter\KVS@SecondOfTwo
272       \fi
273       {%
274          \toks@{#1}%
275       }{%
276          \toks@\expandafter{\the\toks@=#1}%
277       }%
278       \KVS@IfEmpty{#2}{%
279       }{%
280          \KVS@@Equals#2\KVS@Nil
281       }%
282    }%
283 }
```

\KVS@SpaceEquals   Removes spaces before the equals sign.

```
284 \def\KVS@SpaceEquals#1{%
285    \expandafter\KVS@@SpaceEquals\the\toks@#1=\KVS@Nil
286 }
```

\KVS@@SpaceEquals

```
287 \def\KVS@@SpaceEquals#1 =#2\KVS@Nil{%
288   \KVS@IfEmpty{#2}{%
289     \toks@{#1}%
290   }{%
291     \toks@{#1=#2}%
292     \expandafter\KVS@@SpaceEquals\the\toks@\KVS@Nil
293   }%
294 }
```

\KVS@EqualsSpace   Removes spaces after the equals sign.

```
295 \def\KVS@EqualsSpace{%
296   \expandafter\KVS@@EqualsSpace\the\toks@= \KVS@Nil
297 }
```

\KVS@@EqualsSpace

```
298 \def\KVS@@EqualsSpace#1= #2\KVS@Nil{%
299   \KVS@IfEmpty{#2}{%
300     \toks@{#1}%
301   }{%
302     \toks@{#1=#2}%
303     \expandafter\KVS@@EqualsSpace\the\toks@\KVS@Nil
304   }%
305 }
```

3.6   Parsing key value lists

\kv@parse   Normalizes and parses the key value list. Also sets \kv@list.

```
306 \def\kv@parse#1{%
307   \kv@normalize{#1}%
308   \expandafter\kv@parse@normalized\expandafter{\kv@list}%
309 }
```

\kv@parse@normalized   #1: key value list
#2: processor

```
310 \def\kv@parse@normalized#1#2{%
311   \KVS@Parse#1,\KVS@Nil{#2}%
312 }
```

\KVS@Parse   #1,#2: key value list
#3: processor

```
313 \def\KVS@Parse#1,#2\KVS@Nil#3{%
314   \KVS@IfEmpty{#1}{%
315   }{%
316     \KVS@Process#1=\KVS@Nil{#3}%
317   }%
318   \KVS@IfEmpty{#2}{%
319   }{%
320     \KVS@Parse#2\KVS@Nil{#3}%
321   }%
322 }
```

\KVS@Process   #1: key
#2: value, =
#3: processor

```
323 \def\KVS@Process#1=#2\KVS@Nil#3{%
324   \def\kv@key{#1}%
325   \KVS@IfEmpty{#2}{%
326     \let\kv@value\relax
327     #3{#1}{}%
328   }{%
329     \KVS@@Process{#1}#2\KVS@Nil{#3}%
```

```
330    }%
331 }
```

`\KVS@@Process` #1: key
#2: value
#3: processor

```
332 \def\KVS@@Process#1#2=\KVS@Nil#3{%
333 &  \edef\kv@value{\etex@unexpanded{#2}}%
334 $  \begingroup
335 $    \toks@{#2}%
336 $    \xdef\KVS@Global{\the\toks@}%
337 $  \endgroup
338 $  \let\kv@value\KVS@Global
339    #3{#1}{#2}%
340 }
```

3.7 Parsing comma lists

`\comma@parse` Normalizes and parses the key value list. Also sets `\comma@list`.

```
341 \def\comma@parse#1{%
342   \comma@normalize{#1}%
343   \expandafter\comma@parse@normalized\expandafter{\comma@list}%
344 }
```

`\comma@parse@normalized` #1: comma list
#2: processor

```
345 \def\comma@parse@normalized#1#2{%
346   \KVS@CommaParse#1,\KVS@Nil{#2}%
347 }
```

`\KVS@CommaParse` #1,#2: comma list
#3: processor

```
348 \def\KVS@CommaParse#1,#2\KVS@Nil#3{%
349   \KVS@IfEmpty{#1}{%
350   }{%
351     \def\comma@entry{#1}%
352     #3{#1}%
353   }%
354   \KVS@IfEmpty{#2}{%
355   }{%
356     \KVS@CommaParse#2\KVS@Nil{#3}%
357   }%
358 }
```

3.8 Processing key value pairs

`\kv@processor@default`

```
359 \def\kv@processor@default#1#2#3{%
360 &  \unless\ifcsname KV@#1@#2\endcsname
361 $  \begingroup\expandafter\expandafter\expandafter\endgroup
362 $  \expandafter\ifx\csname KV@#1@#2\endcsname\relax
363 &    \unless\ifcsname KVS@#1@handler\endcsname
364 $    \begingroup\expandafter\expandafter\expandafter\endgroup
365 $    \expandafter\ifx\csname KVS@#1@handler\endcsname\relax
366        \kv@error@unknownkey{#1}{#2}%
367      \else
368        \csname KVS@#1@handler\endcsname{#2}{#3}%
369        \relax
370      \fi
371    \else
372      \ifx\kv@value\relax
```

```
373 &        \unless\ifcsname KV@#1@#2@default\endcsname
374 $        \begingroup\expandafter\expandafter\expandafter\endgroup
375 $        \expandafter\ifx\csname KV@#1@#2@default\endcsname\relax
376            \kv@error@novalue{#1}{#2}%
377          \else
378            \csname KV@#1@#2@default\endcsname
379            \relax
380          \fi
381        \else
382          \csname KV@#1@#2\endcsname{#3}%
383        \fi
384   \fi
385 }
```

\kv@set@family@handler

```
386 \def\kv@set@family@handler#1{%
387   \KVS@SetFamilyHandler{#1}\@nil
388 }
```

\KVS@SetFamilyHandler

```
389 \def\KVS@SetFamilyHandler#1\@nil#{%
390   \expandafter\def\csname KVS@#1@handler\endcsname##1##2%
391 }
```

3.9   Error handling

\kv@error@novalue

```
392 \def\kv@error@novalue{%
393   \kv@error@generic{No value specified for}%
394 }
```

\kv@error@unknownkey

```
395 \def\kv@error@unknownkey{%
396   \kv@error@generic{Undefined}%
397 }
```

\kv@error@generic

```
398 \def\kv@error@generic#1#2#3{%
399   \@PackageError{kvsetkeys}{%
400     #1 key '#3'%
401   }{%
402     The keyval family of the key '#3' is '#2'.\MessageBreak
403     \MessageBreak
404     \@ehc
405   }%
406 }
```

3.10   Do it all

\kvsetkeys

```
407 \def\kvsetkeys#1#2{%
408   \kv@parse{#2}{\kv@processor@default{#1}}%
409 }
```

```
410 \KVS@AtEnd
411 ⟨/package⟩
```

4 Test

4.1 Catcode checks for loading

```
412 ⟨*test1⟩
413 \catcode`\{=1 %
414 \catcode`\}=2 %
415 \catcode`\#=6 %
416 \catcode`\@=11 %
417 \expandafter\ifx\csname count@\endcsname\relax
418   \countdef\count@=255 %
419 \fi
420 \expandafter\ifx\csname @gobble\endcsname\relax
421   \long\def\@gobble#1{}%
422 \fi
423 \expandafter\ifx\csname @firstofone\endcsname\relax
424   \long\def\@firstofone#1{#1}%
425 \fi
426 \expandafter\ifx\csname loop\endcsname\relax
427   \expandafter\@firstofone
428 \else
429   \expandafter\@gobble
430 \fi
431 {%
432   \def\loop#1\repeat{%
433     \def\body{#1}%
434     \iterate
435   }%
436   \def\iterate{%
437     \body
438       \let\next\iterate
439     \else
440       \let\next\relax
441     \fi
442     \next
443   }%
444   \let\repeat=\fi
445 }%
446 \def\RestoreCatcodes{}
447 \count@=0 %
448 \loop
449   \edef\RestoreCatcodes{%
450     \RestoreCatcodes
451     \catcode\the\count@=\the\catcode\count@\relax
452   }%
453 \ifnum\count@<255 %
454   \advance\count@ 1 %
455 \repeat
456
457 \def\RangeCatcodeInvalid#1#2{%
458   \count@=#1\relax
459   \loop
460     \catcode\count@=15 %
461   \ifnum\count@<#2\relax
462     \advance\count@ 1 %
463   \repeat
464 }
465 \expandafter\ifx\csname LoadCommand\endcsname\relax
466   \def\LoadCommand{\input kvsetkeys.sty\relax}%
467 \fi
468 \def\Test{%
469   \RangeCatcodeInvalid{0}{47}%
```

```
470  \RangeCatcodeInvalid{58}{64}%
471  \RangeCatcodeInvalid{91}{96}%
472  \RangeCatcodeInvalid{123}{255}%
473  \catcode`\@=12 %
474  \catcode`\\=0 %
475  \catcode`\{=1 %
476  \catcode`\}=2 %
477  \catcode`\#=6 %
478  \catcode`\[=12 %
479  \catcode`\]=12 %
480  \catcode`\%=14 %
481  \catcode`\ =10 %
482  \catcode13=5 %
483  \LoadCommand
484  \RestoreCatcodes
485 }
486 \Test
487 \csname @@end\endcsname
488 \end
489 ⟨/test1⟩
```

4.2   Macro tests

4.2.1   Preamble

```
490 ⟨*test2⟩
491 \NeedsTeXFormat{LaTeX2e}
492 \nofiles
493 \documentclass{article}
494 ⟨noetex⟩\let\SavedUnexpanded\unexpanded
495 ⟨noetex⟩\let\unexpanded\UNDEFINED
496 \makeatletter
497 \chardef\KVS@TestMode=1 %
498 \makeatother
499 \usepackage{kvsetkeys}[2010/01/28]
500 ⟨noetex⟩\let\unexpanded\SavedUnexpanded
501 \usepackage{qstest}
502 \IncludeTests{*}
503 \LogTests{log}{*}{*}
```

4.2.2   Time

```
504 \begingroup\expandafter\expandafter\expandafter\endgroup
505 \expandafter\ifx\csname pdfresettimer\endcsname\relax
506 \else
507   \makeatletter
508   \newcount\SummaryTime
509   \newcount\TestTime
510   \SummaryTime=\z@
511   \newcommand*{\PrintTime}[2]{%
512     \typeout{%
513       [Time #1: \strip@pt\dimexpr\number#2sp\relax\space s]%
514     }%
515 }%
516   \newcommand*{\StartTime}[1]{%
517     \renewcommand*{\TimeDescription}{#1}%
518     \pdfresettimer
519 }%
520   \newcommand*{\TimeDescription}{}%
521   \newcommand*{\StopTime}{%
522     \TestTime=\pdfelapsedtime
523     \global\advance\SummaryTime\TestTime
524     \PrintTime\TimeDescription\TestTime
525 }%
```

```
526  \let\saved@qstest\qstest
527  \let\saved@endqstest\endqstest
528  \def\qstest#1#2{%
529    \saved@qstest{#1}{#2}%
530    \StartTime{#1}%
531  }%
532  \def\endqstest{%
533    \StopTime
534    \saved@endqstest
535  }%
536  \AtEndDocument{%
537    \PrintTime{summary}\SummaryTime
538  }%
539  \makeatother
540 \fi
```

4.2.3   Test sets

```
541 \makeatletter
542 \def\@makeactive#1{%
543   \catcode`#1=13\relax
544 }
545 \@makeactive\,
546 \def,{\errmessage{COMMA}}
547 \@makeother\,
548 \@makeactive\=
549 \def={\errmessage{EQUALS}}
550 \@makeother\=
551
552 \begin{qstest}{normalize}{normalize,active-chars,space-removal}%
553   \def\Test#1#2{%
554     \@makeother\,%
555     \@makeother\=%
556     \scantokens{\toks@={#2}}%
557     \edef\Result{\the\toks@}%
558     \@makeother\,%
559     \@makeother\=%
560     \@Test{#1}%
561     \@makeactive\,%
562     \@Test{#1}%
563     \@makeactive\=%
564     \@Test{#1}%
565     \@makeother\,%
566     \@Test{#1}%
567     \@makeother\=%
568   }%
569   \def\@Test#1{%
570     \scantokens{\kv@normalize{#1}}%
571     \expandafter\expandafter\expandafter\Expect
572     \expandafter\expandafter\expandafter
573     {\expandafter\kv@list\expandafter}\expandafter{\Result}%
574     \Expect*{\ifx\kv@list\Result true\else false\fi}{true}%
575   }%
576   \Test{}{,}%
577   \Test{,}{,}%
578   \Test{,,}{,}%
579   \Test{,,,}{,}%
580   \Test{ , }{,}%
581   \Test{{a}}{,{a},}%
582   \Test{,{a}}{,{a},}%
583   \Test{{a},}{,{a},}%
584   \Test{{a},{b}}{,{a},{b},}%
585   \Test{{b}={c},{}=,{}={},{d}=}{,{b}={c},{}=,{}={},{d}=,}%
586   \Test{{}}{,{},}%
```

```
587    \Test{{},{},{}}{,{},{},{},}%
588    \Test{=}{,=,}%
589    \Test{=,=,=}{,=,=,=,}%
590    \def\TestSet#1{%
591      \Test{#1#1}{,}%
592      \Test{#1#1,#1#1}{,}%
593      \Test{#1#1,#1#1,#1#1}{,}%
594      \Test{#1#1#1#1}{,}%
595      \Test{{a}#1#1=#1#1{b}}{,{a}={b},}%
596    }%
597    \TestSet{ }%
598    \begingroup
599      \let\saved@normalize\kv@normalize
600      \def\kv@normalize#1{%
601        \saved@normalize{#1}%
602        \@onelevel@sanitize\kv@list
603        \@onelevel@sanitize\Result
604      }%
605      \Test{#,#=#,{#}={#},{#}=,{#}}{,#,#=#,{#}={#},{#}=,{#},}%
606    \endgroup
607    \begingroup
608      \def\Test#1#2{%
609        \edef\Result{#2}%
610        \@Test{#1}%
611      }%
612      \Test{{ a = b }}{,{ a = b },}%
613      \@makeactive\,%
614      \Test{{,}}{\string,{\noexpand,}\string,}%
615      \@makeother\,%
616      \@makeactive\=%
617      \Test{a={=}}{,a\string={\noexpand=},}%
618    \endgroup
619    \Test{a=b}{,a=b,}%
620    \Test{a={b}}{,a={b},}%
621    \Test{a ={b}}{,a={b},}%
622    \Test{a= {b}}{,a={b},}%
623    \Test{a = {b}}{,a={b},}%
624    \Test{a = {b} ,}{,a={b},}%
625    \Test{a}{,a,}%
626    \Test{ a}{,a,}%
627    \Test{a }{,a,}%
628    \Test{ a }{,a,}%
629    \Test{, a ,}{,a,}%
630    \Test{, a b ,}{,a b,}%
631    \Test{,a ,}{,a,}%
632    \Test{ a =}{,a=,}%
633    \Test{ a = }{,a=,}%
634    \Test{a =}{,a=,}%
635    \Test{{a} =}{,{a}=,}%
636    \Test{{a}= {}}{,{a}={},}%
637    \Test{, a = {}}{,a={},}%
638    \Test{a,,b}{,a,b,}%
639    \Test{a=\fi}{,a=\fi,}%
640    \Test{a=\iffalse}{,a=\iffalse,}%
641    \Test{a=\iffalse,b=\fi}{,a=\iffalse,b=\fi,}%
642  \end{qstest}
643
644  \begin{qstest}{parse}{parse,brace-removal}
645    \def\Processor#1#2{%
646      \expandafter\Expect\expandafter{\kv@key}{#1}%
647      \toks@{#2}%
648      \edef\x{\the\toks@}%
```

```
649        \ifx\kv@value\relax
650          \Expect*{\the\toks@}{}%
651          \def\Value{<>}%
652        \else
653          \edef\Value{[\the\toks@]}%
654          \@onelevel@sanitize\Value
655        \fi
656        \toks@{#1}%
657        \ifx\Result\@empty
658          \edef\Result{[\the\toks@]=\Value}%
659        \else
660          \edef\Result{\Result,[\the\toks@]=\Value}%
661        \fi
662        \@onelevel@sanitize\Result
663      }%
664      \def\Test#1#2{%
665        \sbox0{%
666          \let\Result\@empty
667          \kv@parse{#1}\Processor
668          \Expect*{\Result}{#2}%
669        }%
670        \Expect*{\the\wd0}{0.0pt}%
671      }%
672      \Test{}{}%
673      \Test{{}}{}%
674      \Test{{{}}}{[]=<>}%
675      \Test{{{{}}}}{[{}]=<>}%
676      \Test{a}{[a]=<>}%
677      \Test{{a}}{[a]=<>}%
678      \Test{{a}}{[a]=<>}%
679      \Test{{{a}}}{[a]=<>}%
680      \Test{{{{a}}}}{[{a}]=<>}%
681      \Test{a=}{[a]=[]}%
682      \Test{{a}=}{[a]=[]}%
683      \Test{{{a}}=}{[{a}]=[]}%
684      \Test{a={}}{[a]=[]}%
685      \Test{{a}={{}}}{[a]=[{}]}%
686      \Test{a=b}{[a]=[b]}%
687      \Test{a=\fi}{[a]=[\fi]}%
688      \Test{a=\iffalse}{[a]=[\iffalse]}%
689      \Test{a=\iffalse,b=\fi}{[a]=[\iffalse],[b]=[\fi]}%
690      \Test{{ a = b }}{[ a ]=[ b ]}%
691      \Test{{{ a = b }}}{[ a = b ]=<>}%
692  \end{qstest}
693
694  \begin{qstest}{comma}{comma,parse}
695      \def\Processor#1{%
696        \expandafter\Expect\expandafter{\comma@entry}{#1}%
697        \toks@{#1}%
698        \ifx\Result\@empty
699          \edef\Result{[\the\toks@]}%
700        \else
701          \edef\Result{\Result,[\the\toks@]}%
702        \fi
703        \@onelevel@sanitize\Result
704      }%
705      \def\Test#1#2{%
706        \sbox0{%
707          \let\Result\@empty
708          \comma@parse{#1}\Processor
709          \Expect*{\Result}{#2}%
710        }%
```

```
711    \Expect*{\the\wd0}{0.0pt}%
712    }%
713  \Test{}{}%
714  \Test{{}}{}%
715  \Test{{{}}}{[{}]}%
716  \Test{a}{[a]}%
717  \Test{{a}}{[a]}%
718  \Test{{{a}}}{[{a}]}%
719  \Test{a=}{[a=]}%
720  \Test{a\fi}{[a\fi]}%
721  \Test{a\iffalse}{[a\iffalse]}%
722  \Test{\iffalse,\fi}{[\iffalse],[\fi]}%
723  \Test{ a , b , c }{[a],[b],[c]}%
724  \Test{ { } ,{ }, { }, { } , { } }{[ ],[ ],[ ],[ ],[ ]}%
725  \Test{ {{}} ,{{}}, {{}}, {{}} , {{}} }{[{}],[{}],[{}],[{}],[{}]}%
726  \end{qstest}
727
728  \begin{document}
729  \end{document}
730  ⟨/test2⟩
```

5 Installation

5.1 Download

Package. This package is available on CTAN[1]:

CTAN:macros/latex/contrib/oberdiek/kvsetkeys.dtx The source file.

CTAN:macros/latex/contrib/oberdiek/kvsetkeys.pdf Documentation.

Bundle. All the packages of the bundle 'oberdiek' are also available in a TDS compliant ZIP archive. There the packages are already unpacked and the documentation files are generated. The files and directories obey the TDS standard.

CTAN:install/macros/latex/contrib/oberdiek.tds.zip

TDS refers to the standard "A Directory Structure for TeX Files" (CTAN:tds/tds.pdf). Directories with `texmf` in their name are usually organized this way.

5.2 Bundle installation

Unpacking. Unpack the `oberdiek.tds.zip` in the TDS tree (also known as `texmf` tree) of your choice. Example (linux):

```
unzip oberdiek.tds.zip -d ~/texmf
```

Script installation. Check the directory `TDS:scripts/oberdiek/` for scripts that need further installation steps. Package **attachfile2** comes with the Perl script `pdfatfi.pl` that should be installed in such a way that it can be called as `pdfatfi`. Example (linux):

```
chmod +x scripts/oberdiek/pdfatfi.pl
cp scripts/oberdiek/pdfatfi.pl /usr/local/bin/
```

5.3 Package installation

Unpacking. The `.dtx` file is a self-extracting **docstrip** archive. The files are extracted by running the `.dtx` through plain-TeX:

```
tex kvsetkeys.dtx
```

---

[1]ftp://ftp.ctan.org/tex-archive/

TDS. Now the different files must be moved into the different directories in your installation TDS tree (also known as `texmf` tree):

```
kvsetkeys.sty            → tex/generic/oberdiek/kvsetkeys.sty
kvsetkeys.pdf            → doc/latex/oberdiek/kvsetkeys.pdf
kvsetkeys-example.tex    → doc/latex/oberdiek/kvsetkeys-example.tex
test/kvsetkeys-test1.tex → doc/latex/oberdiek/test/kvsetkeys-test1.tex
test/kvsetkeys-test2.tex → doc/latex/oberdiek/test/kvsetkeys-test2.tex
test/kvsetkeys-test3.tex → doc/latex/oberdiek/test/kvsetkeys-test3.tex
kvsetkeys.dtx            → source/latex/oberdiek/kvsetkeys.dtx
```

If you have a `docstrip.cfg` that configures and enables `docstrip`'s TDS installing feature, then some files can already be in the right place, see the documentation of `docstrip`.

5.4   Refresh file name databases

If your TeX distribution (teTeX, mikTeX, . . .) relies on file name databases, you must refresh these. For example, teTeX users run `texhash` or `mktexlsr`.

5.5   Some details for the interested

Attached source.   The PDF documentation on CTAN also includes the `.dtx` source file. It can be extracted by AcrobatReader 6 or higher. Another option is `pdftk`, e.g. unpack the file into the current directory:

```
pdftk kvsetkeys.pdf unpack_files output .
```

Unpacking with LaTeX.   The `.dtx` chooses its action depending on the format:

plain-TeX: Run `docstrip` and extract the files.

LaTeX: Generate the documentation.

If you insist on using LaTeX for `docstrip` (really, `docstrip` does not need LaTeX), then inform the autodetect routine about your intention:

```
latex \let\install=y\input{kvsetkeys.dtx}
```

Do not forget to quote the argument according to the demands of your shell.

Generating the documentation.   You can use both the `.dtx` or the `.drv` to generate the documentation. The process can be configured by the configuration file `ltxdoc.cfg`. For instance, put this line into this file, if you want to have A4 as paper format:

```
\PassOptionsToClass{a4paper}{article}
```

An example follows how to generate the documentation with pdfLaTeX:

```
pdflatex kvsetkeys.dtx
makeindex -s gind.ist kvsetkeys.idx
pdflatex kvsetkeys.dtx
makeindex -s gind.ist kvsetkeys.idx
pdflatex kvsetkeys.dtx
```

6   References

[1] A guide to key-value methods, Joseph Wright, second draft for TUG-Boat, 2009-03-17. http://www.texdev.net/wp-content/uploads/2009/03/keyval.pdf

[2] David Carlisle: *The keyval package*; 1999/03/16 v1.13; CTAN:macros/latex/required/graphics/keyval.dtx.

7 History

[2006/03/06 v1.0]

- First version.

[2006/10/19 v1.1]

- Fix of `\kv@set@family@handler`.
- Example added.

[2007/09/09 v1.2]

- Using package infwarerr for error messages.
- Catcode section rewritten.

[2007/09/29 v1.3]

- Normalizing and parsing of comma separated lists added.
- `\kv@normalize` rewritten.
- Robustness increased for normalizing and parsing, e.g. for values with unmatched conditionals.
- ε-TeX is used if available.
- Tests added for normalizing and parsing.

[2009/07/19 v1.4]

- Bug fix for `\kv@normalize`: unwanted space removed (Florent Chervet).

[2009/07/30 v1.5]

- Documentation addition: recommendation for Joseph Wright's review article.

[2009/12/12 v1.6]

- Short info shortened.

[2009/12/22 v1.7]

- Internal optimization (`\KVS@CommaSpace`, ..., `\KVS@EqualsSpace`).

[2010/01/28 v1.8]

- Compatibility to iniTeX added.

8 Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

22