The kvsetkeys package

Heiko Oberdiek
<oberdiek@uni-freiburg.de>

2009/07/30 v1.5

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.

```
⟨*example⟩
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage{kvsetkeys}
\usepackage{keyval}

\makeatletter
\newcommand*{\tag}[2][]{%
  % #1: attributes
  % #2: tag name
  \begingroup
    \toks@={}%
    \let\@endslash\@empty
    \kvsetkeys{tag}{#1}%
    \texttt{%
      \textless #2\the\toks@\@endslash\textgreater
    }%
  \endgroup
}
\kv@set@family@handler{tag}{%
  % #1: key
  % #2: value
  \toks@\expandafter{%
    \the\toks@
    \space
    #1=\string"#2\string"%
  }%
}
\define@key{tag}{/}[]{%
  \def\@endslash{/}%
}
\makeatother

\begin{document}
\begin{tabbing}
  \mbox{}\qquad\=\qquad\=\kill
  \tag{html}\\
  \>\dots\\
  \>\tag[border=1]{table}\\
  \>\>\tag[width=200, span=3, /]{colgroup}\\
  \>\>\dots\\
  \>\tag{/table}\\
```

```
  \>\dots\\
  \tag{/html}\\
\end{tabbing}
\end{document}
⟨/example⟩
```

3 Implementation

3.1 Identification

```
⟨*package⟩
```

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

```
\begingroup
  \catcode44 12 % ,
  \catcode45 12 % -
  \catcode46 12 % .
  \catcode58 12 % :
  \catcode64 11 % @
  \catcode123 1 % {
  \catcode125 2 % }
  \expandafter\let\expandafter\x\csname ver@kvsetkeys.sty\endcsname
  \ifx\x\relax % plain-TeX, first loading
  \else
    \def\empty{}%
    \ifx\x\empty % LaTeX, first loading,
      % variable is initialized, but \ProvidesPackage not yet seen
    \else
      \catcode35 6 % #
      \expandafter\ifx\csname PackageInfo\endcsname\relax
        \def\x#1#2{%
          \immediate\write-1{Package #1 Info: #2.}%
        }%
      \else
        \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
      \fi
      \x{kvsetkeys}{The package is already loaded}%
      \aftergroup\endinput
    \fi
  \fi
\endgroup
```

Package identification:

```
\begingroup
  \catcode35 6 % #
  \catcode40 12 % (
  \catcode41 12 % )
  \catcode44 12 % ,
  \catcode45 12 % -
  \catcode46 12 % .
  \catcode47 12 % /
  \catcode58 12 % :
  \catcode64 11 % @
  \catcode91 12 % [
  \catcode93 12 % ]
  \catcode123 1 % {
  \catcode125 2 % }
  \expandafter\ifx\csname ProvidesPackage\endcsname\relax
    \def\x#1#2#3[#4]{\endgroup
      \immediate\write-1{Package: #3 #4}%
      \xdef#1{#4}%
    }%
  \else
```

```
    \def\x#1#2[#3]{\endgroup
      #2[{#3}]%
      \ifx#1\@undefined
        \xdef#1{#3}%
      \fi
      \ifx#1\relax
        \xdef#1{#3}%
      \fi
    }%
  \fi
\expandafter\x\csname ver@kvsetkeys.sty\endcsname
\ProvidesPackage{kvsetkeys}%
  [2009/07/30 v1.5 Key value parser with default handler support (HO)]
```

```
\begingroup
  \catcode123 1 % {
  \catcode125 2 % }
  \def\x{\endgroup
    \expandafter\edef\csname KVS@AtEnd\endcsname{%
      \catcode35 \the\catcode35\relax
      \catcode64 \the\catcode64\relax
      \catcode123 \the\catcode123\relax
      \catcode125 \the\catcode125\relax
    }%
  }%
\x
\catcode35 6 % #
\catcode64 11 % @
\catcode123 1 % {
\catcode125 2 % }
\def\TMP@EnsureCode#1#2{%
  \edef\KVS@AtEnd{%
    \KVS@AtEnd
    \catcode#1 \the\catcode#1\relax
  }%
  \catcode#1 #2\relax
}
\TMP@EnsureCode{36}{3}% $
\TMP@EnsureCode{38}{4}% &
\TMP@EnsureCode{39}{12}% '
\TMP@EnsureCode{44}{12}% ,
\TMP@EnsureCode{46}{12}% .
\TMP@EnsureCode{47}{12}% /
\TMP@EnsureCode{61}{12}% =
\TMP@EnsureCode{94}{7}% ^ (superscript)
\TMP@EnsureCode{96}{12}% `
\TMP@EnsureCode{126}{13}% ~ (active)
```

3.2 Package loading

```
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname RequirePackage\endcsname\relax
  \input infwarerr.sty\relax
  \input etexcmds.sty\relax
\else
  \RequirePackage{infwarerr}[2007/09/09]%
  \RequirePackage{etexcmds}[2007/09/09]%
\fi
```

3.3 Check for ε-TeX

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

```
\begingroup\expandafter\endgroup
\ifcase0\ifetex@unexpanded
          \expandafter\ifx\csname ifcsname\endcsname\relax
```

```
          \else
            \expandafter\ifx\csname unless\endcsname\relax
            \else
              1%
            \fi
          \fi
        \fi
  \catcode`\$=9 % ignore
  \catcode`\&=14 % comment
\else % e-TeX
  \catcode`\$=14 % comment
  \catcode`\&=9  % ignore
\fi
```

3.4 Generic help macros

\KVS@Empty

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

\KVS@FirstOfTwo

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

\KVS@SecondOfTwo

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

\KVS@IfEmpty

```
\def\KVS@IfEmpty#1{%
& \edef\KVS@Temp{\etex@unexpanded{#1}}%
$ \begingroup
$   \toks@{#1}%
$   \edef\KVS@Temp{\the\toks@}%
$ \expandafter\endgroup
  \ifx\KVS@Temp\KVS@Empty
    \expandafter\KVS@FirstOfTwo
  \else
    \expandafter\KVS@SecondOfTwo
  \fi
}
```

3.5 Normalizing

\kv@normalize

```
\def\kv@normalize#1{%
  \begingroup
    \toks@{,#1,}%
    \KVS@Comma
    \KVS@SpaceComma{ }%
    \KVS@CommaSpace
    \KVS@CommaComma
    \KVS@Equals
    \KVS@SpaceEquals{ }%
    \KVS@EqualsSpace
    \xdef\KVS@Global{\the\toks@}%
  \endgroup
  \let\kv@list\KVS@Global
}
```

\comma@normalize

```
\def\comma@normalize#1{%
  \begingroup
```

```
    \toks@{,#1,}%
    \KVS@Comma
    \KVS@SpaceComma{ }%
    \KVS@CommaSpace
    \KVS@CommaComma
    \xdef\KVS@Global{\the\toks@}%
  \endgroup
  \let\comma@list\KVS@Global
}
```

\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.

```
\begingroup
  \lccode`\,=`\,%
  \lccode`\~=`\,%
\lowercase{\endgroup
  \def\KVS@Comma{%
    \toks@\expandafter{\expandafter}\expandafter
    \KVS@@Comma\the\toks@~\KVS@Nil
  }%
  \def\KVS@@Comma#1~#2\KVS@Nil{%
    \toks@\expandafter{\the\toks@#1}%
    \KVS@IfEmpty{#2}{%
    }{%
      \KVS@@Comma,#2\KVS@Nil
    }%
  }%
}
```

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

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

\KVS@@SpaceComma

```
\def\KVS@@SpaceComma#1 ,#2\KVS@Nil{%
  \KVS@IfEmpty{#2}{%
    \toks@{#1}%
  }{%
    \toks@{#1,#2}%
    \expandafter\KVS@@SpaceComma\the\toks@\KVS@Nil
  }%
}
```

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

```
\def\KVS@CommaSpace{%
  \toks@\expandafter{\the\toks@, }%
  \expandafter\KVS@@CommaSpace\the\toks@\KVS@Nil
}
```

\KVS@@CommaSpace

```
\def\KVS@@CommaSpace#1, #2\KVS@Nil{%
  \KVS@IfEmpty{#2}{%
    \toks@{#1}%
  }{%
    \toks@{#1,#2}%
    \expandafter\KVS@@CommaSpace\the\toks@\KVS@Nil
  }%
}
```

\KVS@CommaComma

Replaces multiple commas by one comma.

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

\KVS@@CommaComma

```
\def\KVS@@CommaComma#1,,#2\KVS@Nil{%
  \toks@{#1,#2}%
  \KVS@IfEmpty{#2}{%
  }{%
    \expandafter\KVS@@CommaComma\the\toks@\KVS@Nil
  }%
}
```

\KVS@Equals

Converts active equals signs into catcode other characters.

```
\begingroup
  \lccode`\==`\=%
  \lccode`\~=`\=%
\lowercase{\endgroup
  \def\KVS@Equals{%
    \toks@\expandafter{\expandafter}\expandafter
    \KVS@@Equals\the\toks@~\KVS@Nil
  }%
  \def\KVS@@Equals#1~#2\KVS@Nil{%
    \edef\KVS@Temp{\the\toks@}%
    \ifx\KVS@Temp\KVS@Empty
      \expandafter\KVS@FirstOfTwo
    \else
      \expandafter\KVS@SecondOfTwo
    \fi
    {%
      \toks@{#1}%
    }{%
      \toks@\expandafter{\the\toks@=#1}%
    }%
    \KVS@IfEmpty{#2}{%
    }{%
      \KVS@@Equals#2\KVS@Nil
    }%
  }%
}
```

\KVS@SpaceEquals

Removes spaces before the equals sign.

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

\KVS@@SpaceEquals

```
\def\KVS@@SpaceEquals#1 =#2\KVS@Nil{%
  \KVS@IfEmpty{#2}{%
    \toks@{#1}%
  }{%
    \toks@{#1=#2}%
    \expandafter\KVS@@SpaceEquals\the\toks@\KVS@Nil
  }%
}
```

\KVS@EqualsSpace

Removes spaces after the equals sign.

```
\def\KVS@EqualsSpace{%
```

```
  \toks@\expandafter{\the\toks@= }%
  \expandafter\KVS@@EqualsSpace\the\toks@\KVS@Nil
}
```

\KVS@@EqualsSpace

```
\def\KVS@@EqualsSpace#1= #2\KVS@Nil{%
  \KVS@IfEmpty{#2}{%
    \toks@{#1}%
  }{%
    \toks@{#1=#2}%
    \expandafter\KVS@@EqualsSpace\the\toks@\KVS@Nil
  }%
}
```

3.6 Parsing key value lists

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

```
\def\kv@parse#1{%
  \kv@normalize{#1}%
  \expandafter\kv@parse@normalized\expandafter{\kv@list}%
}
```

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

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

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

```
\def\KVS@Parse#1,#2\KVS@Nil#3{%
  \KVS@IfEmpty{#1}{%
  }{%
    \KVS@Process#1=\KVS@Nil{#3}%
  }%
  \KVS@IfEmpty{#2}{%
  }{%
    \KVS@Parse#2\KVS@Nil{#3}%
  }%
}
```

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

```
\def\KVS@Process#1=#2\KVS@Nil#3{%
  \def\kv@key{#1}%
  \KVS@IfEmpty{#2}{%
    \let\kv@value\relax
    #3{#1}{}%
  }{%
    \KVS@@Process{#1}#2\KVS@Nil{#3}%
  }%
}
```

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

```
\def\KVS@@Process#1#2=\KVS@Nil#3{%
& \edef\kv@value{\etex@unexpanded{#2}}%
$ \begingroup
$   \toks@{#2}%
```

```
$   \xdef\KVS@Global{\the\toks@}%
$ \endgroup
$ \let\kv@value\KVS@Global
  #3{#1}{#2}%
}
```

3.7 Parsing comma lists

\comma@parse

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

```
\def\comma@parse#1{%
  \comma@normalize{#1}%
  \expandafter\comma@parse@normalized\expandafter{\comma@list}%
}
```

\comma@parse@normalized

#1: comma list
#2: processor

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

\KVS@CommaParse

#1,#2: comma list
#3: processor

```
\def\KVS@CommaParse#1,#2\KVS@Nil#3{%
  \KVS@IfEmpty{#1}{%
  }{%
    \def\comma@entry{#1}%
    #3{#1}%
  }%
  \KVS@IfEmpty{#2}{%
  }{%
    \KVS@CommaParse#2\KVS@Nil{#3}%
  }%
}
```

3.8 Processing key value pairs

\kv@processor@default

```
\def\kv@processor@default#1#2#3{%
& \unless\ifcsname KV@#1@#2\endcsname
$ \begingroup\expandafter\expandafter\expandafter\endgroup
$ \expandafter\ifx\csname KV@#1@#2\endcsname\relax
&   \unless\ifcsname KVS@#1@handler\endcsname
$   \begingroup\expandafter\expandafter\expandafter\endgroup
$   \expandafter\ifx\csname KVS@#1@handler\endcsname\relax
      \kv@error@unknownkey{#1}{#2}%
    \else
      \csname KVS@#1@handler\endcsname{#2}{#3}%
      \relax
    \fi
  \else
    \ifx\kv@value\relax
&     \unless\ifcsname KV@#1@#2@default\endcsname
$     \begingroup\expandafter\expandafter\expandafter\endgroup
$     \expandafter\ifx\csname KV@#1@#2@default\endcsname\relax
        \kv@error@novalue{#1}{#2}%
      \else
        \csname KV@#1@#2@default\endcsname
        \relax
      \fi
    \else
      \csname KV@#1@#2\endcsname{#3}%
```

```
    \fi
  \fi
}
```

\kv@set@family@handler

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

\KVS@SetFamilyHandler

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

3.9 Error handling

\kv@error@novalue

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

\kv@error@unknownkey

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

\kv@error@generic

```
\def\kv@error@generic#1#2#3{%
  \@PackageError{kvsetkeys}{%
    #1 key `#3'%
  }{%
    The keyval family of the key `#3' is `#2'.\MessageBreak
    \MessageBreak
    \@ehc
  }%
}
```

3.10 Do it all

\kvsetkeys

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

```
\KVS@AtEnd
⟨/package⟩
```

4 Test

4.1 Catcode checks for loading

```
⟨*test1⟩
```

```
\catcode`\{=1 %
\catcode`\}=2 %
\catcode`\#=6 %
\catcode`\@=11 %
\expandafter\ifx\csname count@\endcsname\relax
  \countdef\count@=255 %
\fi
```

```
\expandafter\ifx\csname @gobble\endcsname\relax
  \long\def\@gobble#1{}%
\fi
\expandafter\ifx\csname @firstofone\endcsname\relax
  \long\def\@firstofone#1{#1}%
\fi
\expandafter\ifx\csname loop\endcsname\relax
  \expandafter\@firstofone
\else
  \expandafter\@gobble
\fi
{%
  \def\loop#1\repeat{%
    \def\body{#1}%
    \iterate
  }%
  \def\iterate{%
    \body
      \let\next\iterate
    \else
      \let\next\relax
    \fi
    \next
  }%
  \let\repeat=\fi
}%
\def\RestoreCatcodes{}
\count@=0 %
\loop
  \edef\RestoreCatcodes{%
    \RestoreCatcodes
    \catcode\the\count@=\the\catcode\count@\relax
  }%
\ifnum\count@<255 %
  \advance\count@ 1 %
\repeat

\def\RangeCatcodeInvalid#1#2{%
  \count@=#1\relax
  \loop
    \catcode\count@=15 %
  \ifnum\count@<#2\relax
    \advance\count@ 1 %
  \repeat
}
\expandafter\ifx\csname LoadCommand\endcsname\relax
  \def\LoadCommand{\input kvsetkeys.sty\relax}%
\fi
\def\Test{%
  \RangeCatcodeInvalid{0}{47}%
  \RangeCatcodeInvalid{58}{64}%
  \RangeCatcodeInvalid{91}{96}%
  \RangeCatcodeInvalid{123}{255}%
  \catcode`\@=12 %
  \catcode`\\=0 %
  \catcode`\{=1 %
  \catcode`\}=2 %
  \catcode`\#=6 %
  \catcode`\[=12 %
  \catcode`\]=12 %
  \catcode`\%=14 %
  \catcode`\ =10 %
```

```
  \catcode13=5 %
  \LoadCommand
  \RestoreCatcodes
}
\Test
\csname @@end\endcsname
\end
⟨/test1⟩
```

4.2 Macro tests

4.2.1 Preamble

```
⟨*test2⟩
\NeedsTeXFormat{LaTeX2e}
\nofiles
\documentclass{article}
⟨noetex⟩\let\SavedUnexpanded\unexpanded
⟨noetex⟩\let\unexpanded\UNDEFINED
\makeatletter
\chardef\KVS@TestMode=1 %
\makeatother
\usepackage{kvsetkeys}[2009/07/30]
⟨noetex⟩\let\unexpanded\SavedUnexpanded
\usepackage{qstest}
\IncludeTests{*}
\LogTests{log}{*}{*}
```

4.2.2 Time

```
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname pdfresettimer\endcsname\relax
\else
  \makeatletter
  \newcount\SummaryTime
  \newcount\TestTime
  \SummaryTime=\z@
  \newcommand*{\PrintTime}[2]{%
    \typeout{%
      [Time #1: \strip@pt\dimexpr\number#2sp\relax\space s]%
    }%
  }%
  \newcommand*{\StartTime}[1]{%
    \renewcommand*{\TimeDescription}{#1}%
    \pdfresettimer
  }%
  \newcommand*{\TimeDescription}{}%
  \newcommand*{\StopTime}{%
    \TestTime=\pdfelapsedtime
    \global\advance\SummaryTime\TestTime
    \PrintTime\TimeDescription\TestTime
  }%
  \let\saved@qstest\qstest
  \let\saved@endqstest\endqstest
  \def\qstest#1#2{%
    \saved@qstest{#1}{#2}%
    \StartTime{#1}%
  }%
  \def\endqstest{%
    \StopTime
    \saved@endqstest
  }%
  \AtEndDocument{%
    \PrintTime{summary}\SummaryTime
```

```
  }%
  \makeatother
\fi
```

4.2.3 Test sets

```
\makeatletter
\def\@makeactive#1{%
  \catcode`#1=13\relax
}
\@makeactive\,
\def,{\errmessage{COMMA}}
\@makeother\,
\@makeactive\=
\def={\errmessage{EQUALS}}
\@makeother\=

\begin{qstest}{normalize}{normalize,active-chars,space-removal}%
  \def\Test#1#2{%
    \@makeother\,%
    \@makeother\=%
    \scantokens{\toks@={#2}}%
    \edef\Result{\the\toks@}%
    \@makeother\,%
    \@makeother\=%
    \@Test{#1}%
    \@makeactive\,%
    \@Test{#1}%
    \@makeactive\=%
    \@Test{#1}%
    \@makeother\,%
    \@Test{#1}%
    \@makeother\=%
  }%
  \def\@Test#1{%
    \scantokens{\kv@normalize{#1}}%
    \expandafter\expandafter\expandafter\Expect
    \expandafter\expandafter\expandafter
    {\expandafter\kv@list\expandafter}\expandafter{\Result}%
    \Expect*{\ifx\kv@list\Result true\else false\fi}{true}%
  }%
  \Test{}{,}%
  \Test{,}{,}%
  \Test{,,}{,}%
  \Test{,,,}{,}%
  \Test{ , }{,}%
  \Test{{a}}{,{a},}%
  \Test{,{a}}{,{a},}%
  \Test{{a},}{,{a},}%
  \Test{{a},{b}}{,{a},{b},}%
  \Test{{b}={c},{}=,{}={},{d}=}{,{b}={c},{}=,{}={},{d}=,}%
  \Test{{}}{,{},}%
  \Test{{},{},{}}{,{},{},{},}%
  \Test{=}{,=,}%
  \Test{=,=,=}{,=,=,=,}%
  \def\TestSet#1{%
    \Test{#1#1}{,}%
    \Test{#1#1,#1#1}{,}%
    \Test{#1#1,#1#1,#1#1}{,}%
    \Test{#1#1#1#1#1}{,}%
    \Test{{a}#1#1=#1#1{b}}{,{a}={b},}%
  }%
  \TestSet{ }%
  \begingroup
```

```
    \let\saved@normalize\kv@normalize
    \def\kv@normalize#1{%
      \saved@normalize{#1}%
      \@onelevel@sanitize\kv@list
      \@onelevel@sanitize\Result
    }%
    \Test{#,#=#,{#}={#},{#}=,{#}}{,#,#=#,{#}={#},{#}=,{#},}%
  \endgroup
  \begingroup
    \def\Test#1#2{%
      \edef\Result{#2}%
      \@Test{#1}%
    }%
    \Test{{ a = b }}{,{ a = b },}%
    \@makeactive\,%
    \Test{{,}}{\string,{\noexpand,}\string,}%
    \@makeother\,%
    \@makeactive\=%
    \Test{a={=}}{,a\string={\noexpand=},}%
  \endgroup
  \Test{a=b}{,a=b,}%
  \Test{a={b}}{,a={b},}%
  \Test{a ={b}}{,a={b},}%
  \Test{a= {b}}{,a={b},}%
  \Test{a = {b}}{,a={b},}%
  \Test{a = {b} ,}{,a={b},}%
  \Test{a}{,a,}%
  \Test{ a}{,a,}%
  \Test{a }{,a,}%
  \Test{ a }{,a,}%
  \Test{, a ,}{,a,}%
  \Test{, a b ,}{,a b,}%
  \Test{,a ,}{,a,}%
  \Test{ a =}{,a=,}%
  \Test{ a = }{,a=,}%
  \Test{a =}{,a=,}%
  \Test{{a} =}{,{a}=,}%
  \Test{{a}= {}}{,{a}={},}%
  \Test{, a = {}}{,a={},}%
  \Test{a,,b}{,a,b,}%
  \Test{a=\fi}{,a=\fi,}%
  \Test{a=\iffalse}{,a=\iffalse,}%
  \Test{a=\iffalse,b=\fi}{,a=\iffalse,b=\fi,}%
\end{qstest}

\begin{qstest}{parse}{parse,brace-removal}
  \def\Processor#1#2{%
    \expandafter\Expect\expandafter{\kv@key}{#1}%
    \toks@{#2}%
    \edef\x{\the\toks@}%
    \ifx\kv@value\relax
      \Expect*{\the\toks@}{}%
      \def\Value{<>}%
    \else
      \edef\Value{[\the\toks@]}%
      \@onelevel@sanitize\Value
    \fi
    \toks@{#1}%
    \ifx\Result\@empty
      \edef\Result{[\the\toks@]=\Value}%
    \else
      \edef\Result{\Result,[\the\toks@]=\Value}%
```

```
    \fi
    \@onelevel@sanitize\Result
  }%
  \def\Test#1#2{%
    \sbox0{%
      \let\Result\@empty
      \kv@parse{#1}\Processor
      \Expect*{\Result}{#2}%
    }%
    \Expect*{\the\wd0}{0.0pt}%
  }%
  \Test{}{}%
  \Test{{}}{}%
  \Test{{{}}}{[]=<>}%
  \Test{{{{}}}}{[{}]=<>}%
  \Test{a}{[a]=<>}%
  \Test{{a}}{[a]=<>}%
  \Test{{a}}{[a]=<>}%
  \Test{{{a}}}{[a]=<>}%
  \Test{{{{a}}}}{[{a}]=<>}%
  \Test{a=}{[a]=[]}%
  \Test{{a}=}{[a]=[]}%
  \Test{{{a}}=}{[{a}]=[]}%
  \Test{a={}}{[a]=[]}%
  \Test{{a}={{}}}{[a]=[{}]}%
  \Test{a=b}{[a]=[b]}%
  \Test{a=\fi}{[a]=[\fi]}%
  \Test{a=\iffalse}{[a]=[\iffalse]}%
  \Test{a=\iffalse,b=\fi}{[a]=[\iffalse],[b]=[\fi]}%
  \Test{{ a = b }}{[ a ]=[ b ]}%
  \Test{{{ a = b }}}{[ a = b ]=<>}%
\end{qstest}

\begin{qstest}{comma}{comma,parse}
  \def\Processor#1{%
    \expandafter\Expect\expandafter{\comma@entry}{#1}%
    \toks@{#1}%
    \ifx\Result\@empty
      \edef\Result{[\the\toks@]}%
    \else
      \edef\Result{\Result,[\the\toks@]}%
    \fi
    \@onelevel@sanitize\Result
  }%
  \def\Test#1#2{%
    \sbox0{%
      \let\Result\@empty
      \comma@parse{#1}\Processor
      \Expect*{\Result}{#2}%
    }%
    \Expect*{\the\wd0}{0.0pt}%
  }%
  \Test{}{}%
  \Test{{}}{}%
  \Test{{{}}}{[{}]}%
  \Test{a}{[a]}%
  \Test{{a}}{[a]}%
  \Test{{{a}}}{[{a}]}%
  \Test{a=}{[a=]}%
  \Test{a\fi}{[a\fi]}%
  \Test{a\iffalse}{[a\iffalse]}%
  \Test{\iffalse,\fi}{[\iffalse],[\fi]}%
```

```
  \Test{ a , b , c }{[a],[b],[c]}%
  \Test{ { } ,{ }, { }, { } , { } }{[ ],[ ],[ ],[ ],[ ]}%
  \Test{ {{}} ,{{}}, {{}}, {{}} , {{}} }{[{}],[{}],[{}],[{}],[{}]}%
\end{qstest}

\begin{document}
\end{document}
⟨/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
```

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.

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

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.

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.