

The kvsetkeys package

Heiko Oberdiek
<oberdiek@uni-freiburg.de>

2007/09/29 v1.3

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	2
1.1	Motivation	2
1.2	Normalizing key value lists	2
1.3	Parsing key value lists	3
1.4	Processing key value pairs	3
1.5	Default family handler	4
1.6	Put it all together	4
1.7	Comma separated lists	4
2	Example	5
3	Implementation	5
3.1	Identification	5
3.2	Package loading	7
3.3	Check for ε -TeX	7
3.4	Generic help macros	7
3.5	Normalizing	8
3.6	Parsing key value lists	10
3.7	Parsing comma lists	11
3.8	Processing key value pairs	12
3.9	Error handling	12
3.10	Do it all	13
4	Test	13
4.1	Catcode checks for loading	13
4.2	Macro tests	14
4.2.1	Preamble	14
4.2.2	Time	14
4.2.3	Test sets	15
5	Installation	18
5.1	Download	18
5.2	Bundle installation	18
5.3	Package installation	18
5.4	Refresh file name databases	19
5.5	Some details for the interested	19

6	References	19
7	History	19
	[2006/03/06 v1.0]	19
	[2006/10/19 v1.1]	19
	[2007/09/09 v1.2]	19
	[2007/09/29 v1.4]	20
8	Index	20

1 Documentation

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}}}  
\setkeys{fam}{key={{value}}}  
\setkeys{fam}{key= {{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}}  
\kvsetkeys{fam}{key={{value}}}  
\kvsetkeys{fam}{key= {{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 possibilities 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 processor, however, calls a handler instead, if it is provided by the family.

The behaviour in pseudo code:

```

if  $\langle key \rangle$  exists
  call the keyval code of  $\langle key \rangle$ 
else
  if  $\langle handler \rangle$  for  $\langle family \rangle$  exists
     $\langle handler \rangle \{ \langle key \rangle \} \{ \langle value \rangle \}$ 
  else
    raise unknown key error
  fi
fi

```

1.5 Default family handler

`\kv@processor@default` calls $\langle handler \rangle$, 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@handler`:

```
\kv@set@family@handler { $\langle family \rangle$ } { $\langle handler definition \rangle$ }
```

This sets the default family handler for the keyval family $\langle family \rangle$. Inside $\langle handler definition \rangle$ #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 { $\langle family \rangle$ } { $\langle key value list \rangle$ }
```

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

```
\kv@parse { $\langle key value list \rangle$ } { \kv@processor@default { $\langle family \rangle$ } }
```

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 { $\langle comma list \rangle$ }
```

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

```
\comma@parse { $\langle comma list \rangle$ } { $\langle processor \rangle$ }
```

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

```
\comma@parse@normalized { $\langle normalized comma list \rangle$ } { $\langle processor \rangle$ }
```

The list is parsed. Empty entries are ignored. $\langle processor \rangle$ is called for each non-empty entry with the entry as argument:

$\langle processor \rangle \{ \langle entry \rangle \}$

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{} \qqquad = \qqquad = \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 L^AT_EX.

```

49 \begingroup
50 \catcode44 12 % ,
51 \catcode45 12 % -
52 \catcode46 12 % .
53 \catcode58 12 % :
54 \catcode64 11 % @
55 \expandafter\let\expandafter\x\csname ver@kvsetkeys.sty\endcsname
56 \ifcase 0%
57 \ifx\x\relax % plain
58 \else
59 \ifx\x\empty % LaTeX
60 \else
61 1%
62 \fi
63 \fi
64 \else
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 \endgroup
74 \expandafter\endinput
75 \fi
76 \endgroup

```

Package identification:

```

77 \begingroup
78 \catcode40 12 % (
79 \catcode41 12 % )
80 \catcode44 12 % ,
81 \catcode45 12 % -
82 \catcode46 12 % .
83 \catcode47 12 % /
84 \catcode58 12 % :
85 \catcode64 11 % @
86 \expandafter\ifx\csname ProvidesPackage\endcsname\relax
87 \def\x#1#2#3[#4]{\endgroup
88 \immediate\write-1{Package: #3 #4}%
89 \xdef#1{#4}%
90 }%
91 \else
92 \def\x#1#2[#3]{\endgroup
93 #2[#3]}%
94 \ifx#1\relax
95 \xdef#1{#3}%
96 \fi
97 }%
98 \fi
99 \expandafter\x\csname ver@kvsetkeys.sty\endcsname
100 \ProvidesPackage{kvsetkeys}%
101 [2007/09/29 v1.3 Key value parser with default handler support (H0)]
102 \expandafter\edef\csname KVS@AtEnd\endcsname{%
103 \catcode64 \the\catcode64\relax
104 }
105 \catcode64 11 % @
106 \def\TMP@EnsureCode#1#2{%
107 \edef\KVS@AtEnd{%

```

```

108 \KVS@AtEnd
109 \catcode#1 \the\catcode#1\relax
110 }%
111 \catcode#1 #2\relax
112 }
113 \TMP@EnsureCode{36}{3}% $
114 \TMP@EnsureCode{38}{4}% &
115 \TMP@EnsureCode{39}{12}% '
116 \TMP@EnsureCode{44}{12}% ,
117 \TMP@EnsureCode{46}{12}% .
118 \TMP@EnsureCode{47}{12}% /
119 \TMP@EnsureCode{61}{12}% =
120 \TMP@EnsureCode{94}{7}% ^ (superscript)
121 \TMP@EnsureCode{96}{12}% '
122 \TMP@EnsureCode{126}{13}% ~ (active)

```

3.2 Package loading

```

123 \begingroup\expandafter\expandafter\expandafter\endgroup
124 \expandafter\ifx\csname RequirePackage\endcsname\relax
125 \input infwarerr.sty\relax
126 \input etexcmds.sty\relax
127 \else
128 \RequirePackage{infwarerr}[2007/09/09]%
129 \RequirePackage{etexcmds}[2007/09/09]%
130 \fi

```

3.3 Check for ϵ -TeX

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

```

131 \begingroup\expandafter\endgroup
132 \ifcase0\ifetex@unexpanded
133 \expandafter\ifx\csname ifcsname\endcsname\relax
134 \else
135 \expandafter\ifx\csname unless\endcsname\relax
136 \else
137 1%
138 \fi
139 \fi
140 \fi
141 \catcode'\$=9 % ignore
142 \catcode'\&=14 % comment
143 \else % e-TeX
144 \catcode'\$=14 % comment
145 \catcode'\&=9 % ignore
146 \fi

```

3.4 Generic help macros

\KVS@Empty

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

\KVS@FirstOfTwo

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

\KVS@SecondOfTwo

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

\KVS@IfEmpty

```

150 \def\KVS@IfEmpty#1{%
151 & \edef\KVS@Temp{\etex@unexpanded{#1}}%
152 $ \begingroup

```

```

153 $ \toks@{#1}%
154 $ \edef\KVS@Temp{\the\toks@}%
155 $ \expandafter\endgroup
156 \ifx\KVS@Temp\KVS@Empty
157 \expandafter\KVS@FirstOfTwo
158 \else
159 \expandafter\KVS@SecondOfTwo
160 \fi
161 }

```

3.5 Normalizing

\kv@normalize

```

162 \def\kv@normalize#1{%
163 \begingroup
164 \toks@{,#1,}%
165 \KVS@Comma
166 \KVS@SpaceComma{ }%
167 \KVS@CommaSpace
168 \KVS@CommaComma
169 \KVS@Equals
170 \KVS@SpaceEquals{ }%
171 \KVS@EqualsSpace{ }%
172 \xdef\KVS@Global{\the\toks@}%
173 \endgroup
174 \let\kv@list\KVS@Global
175 }

```

\comma@normalize

```

176 \def\comma@normalize#1{%
177 \begingroup
178 \toks@{,#1,}%
179 \KVS@Comma
180 \KVS@SpaceComma{ }%
181 \KVS@CommaSpace
182 \KVS@CommaComma
183 \xdef\KVS@Global{\the\toks@}%
184 \endgroup
185 \let\comma@list\KVS@Global
186 }

```

\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.

```

187 \begingroup
188 \lccode'\,=\,%
189 \lccode'\~==\,%
190 \lowercase{\endgroup
191 \def\KVS@Comma{%
192 \toks@\expandafter{\expandafter}\expandafter
193 \KVS@@Comma\the\toks@\KVS@Nil
194 }%
195 \def\KVS@@Comma#1~#2\KVS@Nil{%
196 \toks@\expandafter{\the\toks@#1}%
197 \KVS@IfEmpty{#2}{%
198 }{%
199 \KVS@@Comma,#2\KVS@Nil
200 }%
201 }%
202 }

```

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


```

203 \def\KVS@SpaceComma#1{%
204   \toks@\expandafter{\the\toks@#1,}%
205   \expandafter\KVS@@SpaceComma\the\toks@\KVS@Nil
206 }

```

\KVS@@SpaceComma

```

207 \def\KVS@@SpaceComma#1 ,#2\KVS@Nil{%
208   \KVS@IfEmpty{#2}{%
209     \toks@{#1}%
210   }{%
211     \toks@{#1,#2}%
212     \expandafter\KVS@@SpaceComma\the\toks@\KVS@Nil
213   }%
214 }

```

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

```

215 \def\KVS@CommaSpace{%
216   \toks@\expandafter{\the\toks@, }%
217   \expandafter\KVS@@CommaSpace\the\toks@\KVS@Nil
218 }

```

\KVS@@CommaSpace

```

219 \def\KVS@@CommaSpace#1, #2\KVS@Nil{%
220   \KVS@IfEmpty{#2}{%
221     \toks@{#1}%
222   }{%
223     \toks@{#1,#2}%
224     \expandafter\KVS@@CommaSpace\the\toks@\KVS@Nil
225   }%
226 }

```

\KVS@CommaComma Replaces multiple commas by one comma.

```

227 \def\KVS@CommaComma{%
228   \toks@\expandafter{\the\toks@,}%
229   \expandafter\KVS@@CommaComma\the\toks@\KVS@Nil
230 }

```

\KVS@@CommaComma

```

231 \def\KVS@@CommaComma#1, ,#2\KVS@Nil{%
232   \toks@{#1,#2}%
233   \KVS@IfEmpty{#2}{%
234   }{%
235     \expandafter\KVS@@CommaComma\the\toks@\KVS@Nil
236   }%
237 }

```

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

```

238 \begingroup
239   \lccode'\=='\=%
240   \lccode'\~='\=%
241 \lowercase{\endgroup
242   \def\KVS@Equals{%
243     \toks@\expandafter{\expandafter}\expandafter
244     \KVS@@Equals\the\toks@\KVS@Nil
245   }%
246   \def\KVS@@Equals#1~#2\KVS@Nil{%
247     \edef\KVS@Temp{\the\toks@}%
248     \ifx\KVS@Temp\KVS@Empty
249       \expandafter\KVS@FirstOfTwo
250     \else
251       \expandafter\KVS@SecondOfTwo

```

```

252 \fi
253 {%
254 \toks@{#1}%
255 }{%
256 \toks@\expandafter{\the\toks@=#1}%
257 }%
258 \KVS@ifEmpty{#2}{%
259 }{%
260 \KVS@@Equals#2\KVS@Nil
261 }%
262 }%
263 }

```

\KVS@SpaceEquals Removes spaces before the equals sign.

```

264 \def\KVS@SpaceEquals#1{%
265 \toks@\expandafter{\the\toks@#1}%
266 \expandafter\KVS@@SpaceEquals\the\toks@\KVS@Nil
267 }

```

\KVS@@SpaceEquals

```

268 \def\KVS@@SpaceEquals#1=#2\KVS@Nil{%
269 \KVS@ifEmpty{#2}{%
270 \toks@{#1}%
271 }{%
272 \toks@{#1=#2}%
273 \expandafter\KVS@@SpaceEquals\the\toks@\KVS@Nil
274 }%
275 }

```

\KVS@EqualsSpace Removes spaces after the equals sign.

```

276 \def\KVS@EqualsSpace{%
277 \toks@\expandafter{\the\toks@= }%
278 \expandafter\KVS@@EqualsSpace\the\toks@\KVS@Nil
279 }

```

\KVS@@EqualsSpace

```

280 \def\KVS@@EqualsSpace#1= #2\KVS@Nil{%
281 \KVS@ifEmpty{#2}{%
282 \toks@{#1}%
283 }{%
284 \toks@{#1=#2}%
285 \expandafter\KVS@@EqualsSpace\the\toks@\KVS@Nil
286 }%
287 }

```

3.6 Parsing key value lists

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

```

288 \def\kv@parse#1{%
289 \kv@normalize{#1}%
290 \expandafter\kv@parse@normalized\expandafter{\kv@list}%
291 }

```

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

```

292 \def\kv@parse@normalized#1#2{%
293 \KVS@Parse#1,\KVS@Nil{#2}%
294 }

```

```

\KVS@Parse #1,#2: key value list
#3: processor
295 \def\KVS@Parse#1,#2\KVS@Nil#3{%
296   \KVS@IfEmpty{#1}{%
297   }{%
298     \KVS@Process#1=\KVS@Nil{#3}%
299   }%
300   \KVS@IfEmpty{#2}{%
301   }{%
302     \KVS@Parse#2\KVS@Nil{#3}%
303   }%
304 }

\KVS@Process #1: key
#2: value, =
#3: processor
305 \def\KVS@Process#1=#2\KVS@Nil#3{%
306   \def\kv@key{#1}%
307   \KVS@IfEmpty{#2}{%
308     \let\kv@value\relax
309     #3{#1}{}%
310   }{%
311     \KVS@@Process{#1}#2\KVS@Nil{#3}%
312   }%
313 }

\KVS@@Process #1: key
#2: value
#3: processor
314 \def\KVS@@Process#1#2=\KVS@Nil#3{%
315 & \edef\kv@value{\etex@unexpanded{#2}}%
316 $ \begingroup
317 $ \toks@{#2}%
318 $ \xdef\KVS@Global{\the\toks@}%
319 $ \endgroup
320 $ \let\kv@value\KVS@Global
321 #3{#1}{#2}%
322 }

```

3.7 Parsing comma lists

```

\comma@parse Normalizes and parses the key value list. Also sets \comma@list.
323 \def\comma@parse#1{%
324   \comma@normalize{#1}%
325   \expandafter\comma@parse@normalized\expandafter{\comma@list}%
326 }

\comma@parse@normalized #1: comma list
#2: processor
327 \def\comma@parse@normalized#1#2{%
328   \KVS@CommaParse#1,\KVS@Nil{#2}%
329 }

\KVS@CommaParse #1,#2: comma list
#3: processor
330 \def\KVS@CommaParse#1,#2\KVS@Nil#3{%
331   \KVS@IfEmpty{#1}{%
332   }{%
333     \def\comma@entry{#1}%
334     #3{#1}%
335   }%

```

```

336 \KVS@ifEmpty{#2}{%
337 }{%
338 \KVS@CommaParse#2\KVS@Nil{#3}%
339 }%
340 }

```

3.8 Processing key value pairs

\kv@processor@default

```

341 \def\kv@processor@default#1#2#3{%
342 & \unless\ifcsname KV@#1@#2\endcsname
343 $ \begingroup\expandafter\expandafter\expandafter\endgroup
344 $ \expandafter\ifx\csname KV@#1@#2\endcsname\relax
345 & \unless\ifcsname KVS@#1@handler\endcsname
346 $ \begingroup\expandafter\expandafter\expandafter\endgroup
347 $ \expandafter\ifx\csname KVS@#1@handler\endcsname\relax
348 \kv@error@unknownkey{#1}{#2}%
349 \else
350 \csname KVS@#1@handler\endcsname{#2}{#3}%
351 \relax
352 \fi
353 \else
354 \ifx\kv@value\relax
355 & \unless\ifcsname KV@#1@#2@default\endcsname
356 $ \begingroup\expandafter\expandafter\expandafter\endgroup
357 $ \expandafter\ifx\csname KV@#1@#2@default\endcsname\relax
358 \kv@error@novalue{#1}{#2}%
359 \else
360 \csname KV@#1@#2@default\endcsname
361 \relax
362 \fi
363 \else
364 \csname KV@#1@#2\endcsname{#3}%
365 \fi
366 \fi
367 }

```

\kv@set@family@handler

```

368 \def\kv@set@family@handler#1{%
369 \KVS@SetFamilyHandler{#1}\@nil
370 }

```

\KVS@SetFamilyHandler

```

371 \def\KVS@SetFamilyHandler#1\@nil{%
372 \expandafter\def\csname KVS@#1@handler\endcsname##1##2%
373 }

```

3.9 Error handling

\kv@error@novalue

```

374 \def\kv@error@novalue{%
375 \kv@error@generic{No value specified for}%
376 }

```

\kv@error@unknownkey

```

377 \def\kv@error@unknownkey{%
378 \kv@error@generic{Undefined}%
379 }

```

\kv@error@generic

```

380 \def\kv@error@generic#1#2#3{%
381   \@PackageError{kvsetkeys}{%
382     #1 key ‘#3’%
383   }{%
384     The keyval family of the key ‘#3’ is ‘#2’.\MessageBreak
385     \MessageBreak
386     \@ehc
387   }%
388 }

```

3.10 Do it all

\kvsetkeys

```

389 \def\kvsetkeys#1#2{%
390   \kv@parse{#2}{\kv@processor@default{#1}}%
391 }

392 \KVS@AtEnd
393 </package>

```

4 Test

4.1 Catcode checks for loading

```

394 <*test1>

395 \catcode'\@=11 %
396 \def\RestoreCatcodes{
397   \count@=0 %
398   \loop
399     \edef\RestoreCatcodes{%
400       \RestoreCatcodes
401       \catcode\the\count@=\the\catcode\count@\relax
402     }%
403   \ifnum\count@<255 %
404     \advance\count@\@ne
405   \repeat
406
407   \def\RangeCatcodeInvalid#1#2{%
408     \count@=#1\relax
409     \loop
410       \catcode\count@=15 %
411       \ifnum\count@<#2\relax
412         \advance\count@\@ne
413       \repeat
414   }
415   \def\Test{%
416     \RangeCatcodeInvalid{0}{47}%
417     \RangeCatcodeInvalid{58}{64}%
418     \RangeCatcodeInvalid{91}{96}%
419     \RangeCatcodeInvalid{123}{255}%
420     \catcode'\@=12 %
421     \catcode'\=0 %
422     \catcode'\{=1 %
423     \catcode'\}=2 %
424     \catcode'\#=6 %
425     \catcode'\[=12 %
426     \catcode'\]=12 %
427     \catcode'\%=14 %
428     \catcode'\ =10 %
429     \catcode13=5 %
430     \input kvsetkeys.sty\relax

```

```

431 \RestoreCatcodes
432 }
433 \Test
434 \csname @@end\endcsname
435 \end
436 </test1>

```

4.2 Macro tests

4.2.1 Preamble

```

437 <*test2>
438 \NeedsTeXFormat{LaTeX2e}
439 \nofiles
440 \documentclass{article}
441 (noetex) \let\SavedUnexpanded\unexpanded
442 (noetex) \let\unexpanded\UNDEFINED
443 \makeatletter
444 \chardef\KVS@TestMode=1 %
445 \makeatother
446 \usepackage{kvsetkeys}[2007/09/29]
447 (noetex) \let\unexpanded\SavedUnexpanded
448 \usepackage{qstest}
449 \IncludeTests{*}
450 \LogTests{log}{*}{*}

```

4.2.2 Time

```

451 \begingroup\expandafter\expandafter\expandafter\endgroup
452 \expandafter\ifx\csname pdfresettimer\endcsname\relax
453 \else
454 \makeatletter
455 \newcount\SummaryTime
456 \newcount\TestTime
457 \SummaryTime=\z@
458 \newcommand*{\PrintTime}[2]{%
459 \typeout{%
460 [Time #1: \strip@pt\dimexpr\number#2sp\relax\space s]%
461 }%
462 }%
463 \newcommand*{\StartTime}[1]{%
464 \renewcommand*{\TimeDescription}{#1}%
465 \pdfresettimer
466 }%
467 \newcommand*{\TimeDescription}{}%
468 \newcommand*{\StopTime}{%
469 \TestTime=\pdfelapsedtime
470 \global\advance\SummaryTime\TestTime
471 \PrintTime\TimeDescription\TestTime
472 }%
473 \let\saved@qstest\qstest
474 \let\saved@endqstest\endqstest
475 \def\qstest#1#2{%
476 \saved@qstest{#1}{#2}%
477 \StartTime{#1}%
478 }%
479 \def\endqstest{%
480 \StopTime
481 \saved@endqstest
482 }%
483 \AtEndDocument{%
484 \PrintTime{summary}\SummaryTime
485 }%
486 \makeatother

```

487 \fi

4.2.3 Test sets

```
488 \makeatletter
489 \def\@makeactive#1{%
490   \catcode'#1=13\relax
491 }
492 \@makeactive\,
493 \def,{\errmessage{COMMA}}
494 \@makeoother\,
495 \@makeactive\=
496 \def={\errmessage{EQUALS}}
497 \@makeoother\=
498
499 \begin{qstest}{normalize}{normalize,active-chars,space-removal}%
500   \def\Test#1#2{%
501     \@makeoother\,%
502     \@makeoother\=%
503     \scantokens{\toks@={#2}}%
504     \edef\Result{\the\toks@}%
505     \@makeoother\,%
506     \@makeoother\=%
507     \@Test{#1}%
508     \@makeactive\,%
509     \@Test{#1}%
510     \@makeactive\=%
511     \@Test{#1}%
512     \@makeoother\,%
513     \@Test{#1}%
514     \@makeoother\=%
515   }%
516   \def\@Test#1{%
517     \scantokens{\kv@normalize{#1}}%
518     \expandafter\expandafter\expandafter\Expect
519     \expandafter\expandafter\expandafter
520     {\expandafter\kv@list\expandafter}\expandafter{\Result}%
521     \Expect*{\ifx\kv@list\Result true\else false\fi}{true}%
522   }%
523   \Test{}{,}%
524   \Test{,}{,}%
525   \Test{,,}{,}%
526   \Test{,,,}{,}%
527   \Test{ , }{,}%
528   \Test{{a}}{,{a},}%
529   \Test{,{a}}{,{a},}%
530   \Test{{a},}{,{a},}%
531   \Test{{a},{b}}{,{a},{b},}%
532   \Test{{b}={c},{},{}={},{d}={},{b}={c},{},{}={},{d}=,}%
533   \Test{{}}{,{},}%
534   \Test{{},{},{}={},{},{}={},{d}=,}%
535   \Test{=}{,=,}%
536   \Test{=,=,=}{,=,=,}%
537   \def\TestSet#1{%
538     \Test{#1#1}{,}%
539     \Test{#1#1,#1#1}{,}%
540     \Test{#1#1,#1#1,#1#1}{,}%
541     \Test{#1#1#1#1#1}{,}%
542     \Test{{a}#1#1=#1#1{b}}{,{a}={b},}%
543   }%
544   \TestSet{ }%
545   \begin{group
546     \let\saved@normalize\kv@normalize
547     \def\kv@normalize#1{%
```

```

548     \saved@normalize{#1}%
549     \@onelevel@sanitize\kv@list
550     \@onelevel@sanitize\Result
551   }%
552   \Test{#,#=#,{#}={#},{#}=#,{#}}{#,#=#,{#}={#},{#}=#,{#},}%
553 \endgroup
554 \begingroup
555   \def\Test#1#2{%
556     \edef\Result{#2}%
557     \@Test{#1}%
558   }%
559   \Test{{ a = b }}{,{ a = b },}%
560   \@makeactive\,%
561   \Test{{,}}{\string,{\noexpand,}\string,}%
562   \@makeother\,%
563   \@makeactive\=%
564   \Test{a={}}{,a\string={\noexpand=},}%
565 \endgroup
566 \Test{a=b}{,a=b,}%
567 \Test{a={b}}{,a={b},}%
568 \Test{a = {b}}{,a={b},}%
569 \Test{a= {b}}{,a={b},}%
570 \Test{a = {b}}{,a={b},}%
571 \Test{a = {b} ,}{,a={b},}%
572 \Test{a}{,a,}%
573 \Test{ a}{,a,}%
574 \Test{a }{,a,}%
575 \Test{ a }{,a,}%
576 \Test{, a ,}{,a,}%
577 \Test{, a b ,}{,a b,}%
578 \Test{,a ,}{,a,}%
579 \Test{ a =}{,a=,}%
580 \Test{ a = }{,a=,}%
581 \Test{a =}{,a=,}%
582 \Test{{a} =}{,{a}=,}%
583 \Test{{a}= }{,{a}={},}%
584 \Test{, a = }{,a={},}%
585 \Test{a,,b}{,a,b,}%
586 \Test{a=\fi}{,a=\fi,}%
587 \Test{a=\iffalse}{,a=\iffalse,}%
588 \Test{a=\iffalse,b=\fi}{,a=\iffalse,b=\fi,}%
589 \end{qstest}
590
591 \begin{qstest}{parse}{parse,brace-removal}
592   \def\Processor#1#2{%
593     \expandafter\Expect\expandafter{\kv@key}{#1}%
594     \toks@{#2}%
595     \edef\x{\the\toks@}%
596     \ifx\kv@value\relax
597       \Expect*{\the\toks@}{}%
598       \def\Value{<>}%
599     \else
600       \edef\Value{[\the\toks@]}%
601       \@onelevel@sanitize\Value
602       \fi
603       \toks@{#1}%
604       \ifx\Result\@empty
605         \edef\Result{[\the\toks@]=\Value}%
606       \else
607         \edef\Result{\Result,[\the\toks@]=\Value}%
608       \fi
609       \@onelevel@sanitize\Result

```



```

610 }%
611 \def\Test#1#2{%
612   \let\Result\@empty
613   \kv@parse{#1}\Processor
614   \Expect*\Result{#2}%
615 }%
616 \Test{}{}%
617 \Test{{}}{}%
618 \Test{{{}}}{[]=<>}%
619 \Test{{{}}}{[{}]=<>}%
620 \Test{a}{[a]=<>}%
621 \Test{{a}}{[a]=<>}%
622 \Test{{a}}{[a]=<>}%
623 \Test{{{a}}}{[a]=<>}%
624 \Test{{{a}}}{[[a]]=<>}%
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=\fi}{[a]=[\fi]}%
632 \Test{a=\iffalse}{[a]=[\iffalse]}%
633 \Test{a=\iffalse,b=\fi}{[a]=[\iffalse],[b]=[\fi]}%
634 \Test{{ a = b }}{[ a ]=[ b ]}%
635 \Test{{{ a = b }}}{[ a = b ]=<>}%
636 \end{qstest}
637
638 \begin{qstest}{comma}{comma,parse}
639   \def\Processor#1{%
640     \expandafter\Expect\expandafter{\comma@entry}{#1}%
641     \toks@{#1}%
642     \ifx\Result\@empty
643       \edef\Result{[\the\toks@]}%
644     \else
645       \edef\Result{\Result,[\the\toks@]}%
646     \fi
647     \@onelevel@sanitize\Result
648   }%
649   \def\Test#1#2{%
650     \let\Result\@empty
651     \comma@parse{#1}\Processor
652     \Expect*\Result{#2}%
653   }%
654 \tracingmacros=1
655 \Test{}{}%
656 \Test{{}}{}%
657 \Test{{{}}}{[{}]}%
658 \Test{a}{[a]}%
659 \Test{{a}}{[a]}%
660 \Test{{{a}}}{[[a]]}%
661 \Test{a=}{[a=]}%
662 \Test{a\fi}{[a\fi]}%
663 \Test{a\iffalse}{[a\iffalse]}%
664 \Test{\iffalse,\fi}{[\iffalse],[\fi]}%
665 \Test{ a , b , c }{[a],[b],[c]}%
666 \Test{ { } , { } , { } , { } , { } }{[ ],[ ],[ ],[ ],[ ]}%
667 \Test{ {{}} , {{}} , {{}} , {{}} , {{}} }{[{}],[{}],[{}],[{}],[{}]}%
668 \end{qstest}
669
670 \begin{document}
671 \end{document}

```

5 Installation

5.1 Download

Package. This package is available on CTAN¹:

`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:macros/latex/contrib/oberdiek/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):

<code>kvsetkeys.sty</code>	→ <code>tex/generic/oberdiek/kvsetkeys.sty</code>
<code>kvsetkeys.pdf</code>	→ <code>doc/latex/oberdiek/kvsetkeys.pdf</code>
<code>kvsetkeys-example.tex</code>	→ <code>doc/latex/oberdiek/kvsetkeys-example.tex</code>
<code>kvsetkeys-test1.tex</code>	→ <code>doc/latex/oberdiek/kvsetkeys-test1.tex</code>
<code>kvsetkeys-test2.tex</code>	→ <code>doc/latex/oberdiek/kvsetkeys-test2.tex</code>
<code>kvsetkeys-test3.tex</code>	→ <code>doc/latex/oberdiek/kvsetkeys-test3.tex</code>
<code>kvsetkeys.dtx</code>	→ <code>source/latex/oberdiek/kvsetkeys.dtx</code>

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<http://ftp.ctan.org/tex-archive/>

5.4 Refresh file name databases

If your $\text{T}_{\text{E}}\text{X}$ distribution ($\text{t}_{\text{e}}\text{T}_{\text{E}}\text{X}$, $\text{m}_{\text{i}}\text{k}_{\text{T}}\text{E}_{\text{X}}$, ...) relies on file name databases, you must refresh these. For example, $\text{t}_{\text{e}}\text{T}_{\text{E}}\text{X}$ users run `texhash` or `mktextlsr`.

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 $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$. The `.dtx` chooses its action depending on the format:

plain- $\text{T}_{\text{E}}\text{X}$: Run `docstrip` and extract the files.

$\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$: Generate the documentation.

If you insist on using $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ for `docstrip` (really, `docstrip` does not need $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$), 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 `pdf $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$` :

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

6 References

- [1] 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 v1.4]

- 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.

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.

Symbols	
<code>\#</code>	424
<code>\\$</code>	141, 144
<code>\%</code>	427
<code>\&</code>	142, 145
<code>\,</code>	188, 189, 492, 494, 501, 505, 508, 512, 560, 562
<code>\=</code>	36, 239, 240, 495, 497, 502, 506, 510, 514, 563
<code>\></code>	38, 39, 40, 41, 42, 43
<code>\@</code>	395, 420
<code>\@PackageError</code>	381
<code>\@Test</code>	507, 509, 511, 513, 516, 557
<code>\@ehc</code>	386
<code>\@empty</code>	13, 604, 612, 642, 650
<code>\@endslash</code>	13, 16, 30
<code>\@makeactive</code>	489, 492, 495, 508, 510, 560, 563
<code>\@makeoother</code>	494, 497, 501, 502, 505, 506, 512, 514, 562
<code>\@ne</code>	404, 412
<code>\@nil</code>	369, 371
<code>\@onelevel@sanitize</code>	549, 550, 601, 609, 647
<code>\[</code>	425
<code>\[</code> ...	37, 38, 39, 40, 41, 42, 43, 44, 421
<code>\{</code>	422
<code>\}</code>	423
<code>\]</code>	426
<code>\~</code>	189, 240
<code>_</code>	428
A	
<code>\advance</code>	404, 412, 470
<code>\AtEndDocument</code>	483
B	
<code>\begin</code>	34, 35, 499, 591, 638, 670
C	
<code>\catcode</code> 50, 51, 52, 53, 54, 78, 79, 80, 81, 82, 83, 84, 85, 103, 105, 109,	
111, 141, 142, 144, 145, 395, 401, 410, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 490	
<code>\chardef</code>	444
<code>\comma@entry</code>	333, 640
<code>\comma@list</code>	185, 325
<code>\comma@normalize</code>	4, 176, 324
<code>\comma@parse</code>	4, 323, 651
<code>\comma@parse@normalized</code> .	4, 325, 327
<code>\count@</code>	397, 401, 403, 404, 408, 410, 411, 412
<code>\csname</code>	55, 65, 86, 99, 102, 124, 133, 135, 344, 347, 350, 357, 360, 364, 372, 434, 452
D	
<code>\define@key</code>	29
<code>\dimexpr</code>	460
<code>\documentclass</code>	2, 440
<code>\dots</code>	38, 41, 43
E	
<code>\empty</code>	59
<code>\end</code>	45, 46, 435, 589, 636, 668, 671
<code>\endcsname</code>	55, 65, 86, 99, 102, 124, 133, 135, 342, 344, 345, 347, 350, 355, 357, 360, 364, 372, 434, 452
<code>\endinput</code>	74
<code>\endqstest</code>	474, 479
<code>\errmessage</code>	493, 496
<code>\etex@unexpanded</code>	151, 315
<code>\Expect</code> 518, 521, 593, 597, 614, 640, 652	
I	
<code>\ifcase</code>	56, 132
<code>\ifcsname</code>	342, 345, 355
<code>\ifetex@unexpanded</code>	132
<code>\iffalse</code> ..	587, 588, 632, 633, 663, 664
<code>\ifnum</code>	403, 411
<code>\ifx</code>	57, 59, 65, 86, 94, 124, 133, 135, 156, 248, 344, 347, 354, 357, 452, 521, 596, 604, 642
<code>\immediate</code>	67, 88

\IncludeTests	449	N	
\input	125, 126, 430	\NeedsTeXFormat	438
		\newcommand	8, 458, 463, 467, 468
K		\newcount	455, 456
\kill	36	\nofiles	439
\kv@error@generic	375, 378, 380	\number	460
\kv@error@novalue	358, 374		
\kv@error@unknownkey	348, 377	P	
\kv@key	306, 593	\PackageInfo	70
\kv@list	174, 290, 520, 521, 549	\pdfelapsedtime	469
\kv@normalize	2, 162, 289, 517, 546, 547	\pdfresettimer	465
\kv@parse	3, 288, 390, 613	\PrintTime	458, 471, 484
\kv@parse@normalized	3, 290, 292	\Processor	592, 613, 639, 651
\kv@processor@default	3, 341, 390	\ProvidesPackage	100
\kv@set@family@handler	4, 20, 368		
\kv@value	308, 315, 320, 354, 596	Q	
\KVS@@Comma	193, 195, 199	\qqquad	36
\KVS@@CommaComma	229, 231	\qstest	473, 475
\KVS@@CommaSpace	217, 219		
\KVS@@Equals	244, 246, 260	R	
\KVS@@EqualsSpace	278, 280	\RangeCatcodeInvalid	
\KVS@@Process	311, 314		407, 416, 417, 418, 419
\KVS@@SpaceComma	205, 207	\renewcommand	464
\KVS@@SpaceEquals	266, 268	\repeat	405, 413
\KVS@AtEnd	107, 108, 392	\RequirePackage	128, 129
\KVS@Comma	165, 179, 187	\RestoreCatcodes	396, 399, 400, 431
\KVS@CommaComma	168, 182, 227	\Result	504, 520, 521, 550,
\KVS@CommaParse	328, 330		556, 604, 605, 607, 609, 612,
\KVS@CommaSpace	167, 181, 215		614, 642, 643, 645, 647, 650, 652
\KVS@Empty	147, 156, 248	S	
\KVS@Equals	169, 238	\saved@endqstest	474, 481
\KVS@EqualsSpace	171, 276	\saved@normalize	546, 548
\KVS@FirstOfTwo	148, 157, 249	\saved@qstest	473, 476
\KVS@Global	172, 174, 183, 185, 318, 320	\SavedUnexpanded	441, 447
\KVS@IfEmpty		\scantokens	503, 517
	150, 197, 208, 220, 233, 258,	\space	25, 460
	269, 281, 296, 300, 307, 331, 336	\StartTime	463, 477
\KVS@Nil	193, 195, 199, 205, 207, 212,	\StopTime	468, 480
	217, 219, 224, 229, 231, 235,	\strip@pt	460
	244, 246, 260, 266, 268, 273,	\SummaryTime	455, 457, 470, 484
	278, 280, 285, 293, 295, 298,		
	302, 305, 311, 314, 328, 330, 338	T	
\KVS@Parse	293, 295	\tag	8, 37, 39, 40, 42, 44
\KVS@Process	298, 305	\Test	415, 433, 500, 523, 524, 525,
\KVS@SecondOfTwo	149, 159, 251		526, 527, 528, 529, 530, 531,
\KVS@SetFamilyHandler	369, 371		532, 533, 534, 535, 536, 538,
\KVS@SpaceComma	166, 180, 203		539, 540, 541, 542, 552, 555,
\KVS@SpaceEquals	170, 264		559, 561, 564, 566, 567, 568,
\KVS@Temp	151, 154, 156, 247, 248		569, 570, 571, 572, 573, 574,
\KVS@TestMode	444		575, 576, 577, 578, 579, 580,
\kvsetkeys	4, 14, 389		581, 582, 583, 584, 585, 586,
			587, 588, 611, 616, 617, 618,
L			619, 620, 621, 622, 623, 624,
\lccode	188, 189, 239, 240		625, 626, 627, 628, 629, 630,
\LogTests	450		631, 632, 633, 634, 635, 649,
\loop	398, 409		655, 656, 657, 658, 659, 660,
\lowercase	190, 241		661, 662, 663, 664, 665, 666, 667
		\TestSet	537, 544
M		\TestTime	456, 469, 470, 471
\makeatletter	7, 443, 454, 488	\textgreater	16
\makeatother	32, 445, 486	\textless	16
\mbox	36	\texttt	15
\MessageBreak	384, 385		

<code>\the</code>	16, 24, 103, 109, 154, 172, 183, 193, 196, 204, 205, 212, 216, 217, 224, 228, 229, 235, 244, 247, 256, 265, 266, 273, 277, 278, 285, 318, 401, 504, 595, 597, 600, 605, 607, 643, 645	<code>\typeout</code>	459
<code>\TimeDescription</code>	464, 467, 471	U	
<code>\TMP@EnsureCode</code> 106, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122		<code>\UNDEFINED</code>	442
<code>\toks@</code> . 12, 16, 23, 24, 153, 154, 164, 172, 178, 183, 192, 193, 196, 204, 205, 209, 211, 212, 216, 217, 221, 223, 224, 228, 229, 232, 235, 243, 244, 247, 254, 256, 265, 266, 270, 272, 273, 277, 278, 282, 284, 285, 317, 318, 503, 504, 594, 595, 597, 600, 603, 605, 607, 641, 643, 645		<code>\unexpanded</code>	441, 442, 447
<code>\tracingmacros</code>	654	<code>\unless</code>	342, 345, 355
		<code>\usepackage</code>	3, 4, 5, 446, 448
		V	
		<code>\Value</code>	598, 600, 601, 605, 607
		W	
		<code>\write</code>	67, 88
		X	
		<code>\x</code> 55, 57, 59, 66, 70, 72, 87, 92, 99, 595	
		Z	
		<code>\z@</code>	457