

The ldcsetup package*

DPC, FMi

2009/05/06

Abstract

Basic code shared by several other packages currently under development.

```
1 <*package>
2 \ProvidesExplPackage
3   {\filename}{\filedate}{\fileversion}{\filedescription}
4 \RequirePackage{keyval}
5 \RequirePackage{expl3}
6 \ExplSyntaxOff
```

1 Ignoring White Space

Within a package, or in a document preamble, you can use `\IgnoreWhiteSpace` this makes white space ignored, and `@` a letter, and `~` a space character. It lasts until the end of the current package or preamble.

Puts an ignored space at the end of the line, so `'\ '` does the right thing. (Perhaps should instead just change catcode of `~` to `M`)

```
7 \def\IgnoreWhiteSpace{%
8   \edef\@tempa{%
9     \catcode'\noexpand\~= \the\catcode'\~\relax
10    \catcode'\noexpand\ = \the\catcode'\ \relax
11    \catcode'\noexpand\^^I= \the\catcode'\^^I\relax
12    \catcode'\noexpand\@= \the\catcode'\@\relax
13    \endlinechar= \the\endlinechar\relax
14  }%
15  \ifx\@currname\@empty
16    \expandafter\AtBeginDocument\expandafter{\@tempa}%
17  \else
18    \expandafter\AtEndOfPackage\expandafter{\@tempa}%
19  \fi
20  \catcode'\~=10\relax
21  \catcode'\ =9\relax
22  \catcode'\^^I=9\relax
```

*This file has version number 1295, last revised 2009/05/06.

```

23 \makeatletter
24 \endlinechar=' %
25 \relax
26 }

```

The new syntax requires : and _ to be letters; instead of putting that into \IgnoreWhiteSpace we make a new command which helps to gradually change packages from old to new syntax.

```

27 \def\InternalSyntaxOn{%
28   \edef\@tempa{%
29     \catcode'\noexpand\~=\the\catcode'\~\relax
30     \catcode'\noexpand\ =\the\catcode'\ \relax
31     \catcode'\noexpand\^^I=\the\catcode'\^^I\relax
32     \catcode'\noexpand\@=\the\catcode'\@\relax
33     \catcode'\noexpand\:=\the\catcode'\:\relax
34     \catcode'\noexpand\_=\the\catcode'\_\relax
35     \endlinechar=\the\endlinechar\relax
36 %Fmi fix is not properly reset
37   \endlinechar=13\relax
38   }%
39 %%% ???
40 % this needs further work, don't think it is working
41   \ifx\InternalSyntaxOff\relax
42     \expandafter\def\expandafter\InternalSyntaxOff\expandafter
43       {\@tempa\let\InternalSyntaxOff\relax}%
44   \fi
45   \ifx\@currname\@empty
46     \expandafter\AtBeginDocument\expandafter{\@tempa}%
47   \else
48     \expandafter\AtEndOfPackage\expandafter{\@tempa}%
49   \fi
50   \catcode'\~ =10\relax
51   \catcode'\ =9\relax
52   \catcode'\^^I=9\relax
53   \makeatletter
54   \catcode'\_ =11\relax
55   \catcode'\:=11\relax
56   \endlinechar=' %
57   \relax
58 }
59 \let\InternalSyntaxOff\relax

Do it for this package.

60 %\InternalSyntaxOn
61 \ExplSyntaxOn

```

2 KV fixes

Allow¹ key names to start with \.

```
62 \cs_set_nopar:Npn \define@key#1#2{%
63   \@ifnextchar[
64     {\KV@def{#1}{\string#2}}
65     {\@namedef{KV@#1\string#2}####1}}
66 \cs_set_nopar:Npn \setkeys#1#2{%
67   \cs_set_nopar:Npn \KV@prefix{KV@#1\expandafter\string}%
68   \KV@do#2,\relax,}
```

Make KV # safe.

```
69 \newtoks\KV@toks
70 \cs_set_nopar:Npn \KV@@sp@c#1\@nil#2\relax#3{\KV@toks{#1}\cs_set_nopar:Npx #3{\the\KV@toks}}
```

Generate error messages on missing ‘,’.

More exactly if two ‘=’ appear after a key generate an error. The current KV just silently ignores everything after the second ‘=’.

```
71 \cs_set_nopar:Npn \KV@equal{=}
72 \cs_set_nopar:Npn \KV@split#1=#2=#3\relax{%
73   \KV@@sp@def\@tempa{#1}%
74   \cs_set_nopar:Npn \@tempd{#3}%
75   \ifx\@tempa\@empty\else
76     \expandafter\cs_set_eq:NN \expandafter\@tempc
77     \csname\KV@prefix\@tempa\endcsname
78     \ifx\@tempc\relax
79       \KV@error{\@tempa\space \expandafter\@gobbletwo\string\@undefined}\@eha
80     \else
81       \ifx\@tempd\@empty
82         \KV@default
83       \else
84         \KV@@sp@def\@tempb{#2}%
85         \ifx\@tempd\KV@equal
86           \expandafter\@tempc\expandafter{\@tempb}\relax
87         \else
88           \KV@error{Extra~==~ sign~ after~ ‘#1’}\KV@erry
89         \fi
90       \fi
91     \fi
92   \fi}
93 \cs_set_nopar:Npn \KV@erry{\expandafter\KV@errx\meaning\@tempd\relax ignored\MessageBreak
94   missing~comma~in~‘\expandafter\strip@prefix\meaning\@tempb’~?}
95 \cs_set_nopar:Npn \KV@errx#1>#2==\relax{%
96   ‘#2’ }
```

¹Not needed now?

```

97 \cs_set_nopar:Npn \KV@default{%
98   \expandafter\cs_set_eq:NN \expandafter\@tempb
99   \csname\KV@prefix\@tempa @default\endcsname
100  \ifx\@tempb\relax
101    \KV@error{No value specified for \@tempa}\@eha
102  \else
103    \@tempb\relax
104  \fi}

```

LaTeX style error message.

```

105 \cs_set_nopar:Npn \KV@error#1#2{\PackageError{keyval}{#1}{#2}}

```

Instead of doing a full KV parse, and evaluating all the keys, you might want to parse a parameter list, even for undefined keys, removing spaces, splitting up the ‘,’ and ‘=’ and resolving cases where no value is supplied. This is used in `\DeclareGenericFunction` which *defines* a set of keys via such a KV parse.

A setting of , key = value , will result in `\KV@elt{key}{value}` being added to the list, a setting of , key , will result in `\KV@default@elt{key}` being added. At the end of the parse the list is executed. No keys are checked at this stage (so no csnames are used up) the two `\KV...@elt` commands must be defined as appropriate.

```

106 \cs_set_nopar:Npn \KV@parse#1{
107   \begingroup

```

Locally fudge `\KV@@def` to just add to a token register rather than make a definition.

```

108 \cs_set_nopar:Npn \KV@@sp@c##1\@nil##2\relax##3{\addto@hook##3{\##1}}

```

Locally fudge `\KV@split` to just add to a token register rather than execute the code for the found keys.

```

109 \cs_set_nopar:Npn \KV@split##1=##2=##3\relax{%
110   \cs_set_nopar:Npn \@tempd{##3}%
111   \expandafter\ifx\expandafter=\@firstofone##1=\else
112     \ifx\@tempd\@empty
113       \addto@hook\KV@toks\KV@default@elt
114       \KV@@sp@def\KV@toks{##1}%
115     \else
116       \ifx\@tempd\KV@equal
117         \addto@hook\KV@toks\KV@elt
118         \KV@@sp@def\KV@toks{##1}%
119         \KV@@sp@def\KV@toks{##2}%
120       \else
121         \KV@err{Extra ‘=’ after ‘##1’}\@ehd
122       \fi
123   \fi
124 \fi}

```

Initialise

```

125 \KV@toks{}

```

Do the parse

```
126 \KV@do#1,\relax,
```

Evaluate the token register outside the group.

```
127 \expandafter
```

```
128 \endgroup
```

```
129 \the\KV@toks}
```

3 Calc fixes

Count Register assignments via calc. Here we also provide some global versions since a normal global prefix won't have any effect.

```
130 \cs_set_eq:NN \SetInternalCounter \calc_int_set:Nn
```

```
131 \cs_set_eq:NN \GSetInternalCounter\calc_int_gset:Nn
```

4 Misc code that is best collected in one place

This section will receive odd code that would be of potential use to other packages and should eventually made it to the kernel (perhaps).

`\UndeclareRobustCommand` Remove a robust command from memory if it isn't used any longer. Of course this doesn't free the hash table but at least the space gets reclaimed.

```
132 \cs_set_nopar:Npn \UndeclareRobustCommand#1{%
```

```
133 \cs_gundefine:N #1
```

```
134 \exp_args:Nc \cs_gundefine:N {\cs_to_str:N #1~}
```

```
135 }
```

```
136 </package>
```