

The `ldcsetup` package*

DPC, FMi

2007/09/12

Abstract

Basic code shared by several other packages currently under development.

```
1 <*package>
2 \ProvidesExplPackage
3   {\filename}{\filedate}{\fileversion}{\filedescription}
4 \RequirePackage{keyval,l3calc}
5 \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 `~`)

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

*This file has version number 636, last revised 2007/09/12.

```

23 \endlinechar=' %
24 \relax
25 }

```

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.

```

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

```

Do it for this package.

```

59 %\InternalSyntaxOn
60 \ExplSyntaxOn

```

2 KV fixes

Allow¹ key names to start with \.

¹Not needed now?

```

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

    Make KV # safe.

68 \newtoks\KV@toks
69 \def\KV@sp@c#1\@nil#2\relax#3{\KV@toks{#1}\edef#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 '='.
70 \def\KV@equal{=}

71 \def\KV@split#1=#2=#3\relax{%
72   \KV@sp@def\@tempa{#1}%
73   \def\@tempd{#3}%
74   \ifx\@tempa\@empty\else
75     \expandafter\let\expandafter\@tempc
76     \csname\KV@prefix\@tempa\endcsname
77     \ifx\@tempc\relax
78       \KV@error{\@tempa\space \expandafter\@gobbletwo\string\@undefined}\@eha
79     \else
80       \ifx\@tempd\@empty
81         \KV@default
82       \else
83         \KV@sp@def\@tempb{#2}%
84         \ifx\@tempd\KV@equal
85           \expandafter\@tempc\expandafter{\@tempb}\relax
86         \else
87           \KV@error{Extra~==~ sign~ after~ '#1'}\KV@erry
88         \fi
89       \fi
90     \fi
91   \fi}

92 \def\KV@erry{\expandafter\KV@errx\meaning\@tempd\relax ignored\MessageBreak
93             missing~comma~in~'\expandafter\strip@prefix\meaning\@tempb'~?}

94 \def\KV@errx#1>#2==\relax{%
95   '#2' }

96 \def\KV@default{%
97   \expandafter\let\expandafter\@tempb
98   \csname\KV@prefix\@tempa @default\endcsname
99   \ifx\@tempb\relax
100     \KV@error{No value specified for \@tempa}\@eha
101   \else
102     \@tempb\relax

```

103 \fi}

LaTeX style error message.

104 \def\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.

105 \def\KV@parse#1{

106 \begingroup

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

107 \def\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.

108 \def\KV@split##1=##2=##3\relax{%

109 \def\@tempd{##3}%

110 \expandafter\ifx\expandafter=\@firstofone##1=\else

111 \ifx\@tempd\@empty

112 \addto@hook\KV@toks\KV@default@elt

113 \KV@sp@def\KV@toks{##1}%

114 \else

115 \ifx\@tempd\KV@equal

116 \addto@hook\KV@toks\KV@elt

117 \KV@sp@def\KV@toks{##1}%

118 \KV@sp@def\KV@toks{##2}%

119 \else

120 \KV@err{Extra ‘=’ after ‘##1’}\@ehd

121 \fi

122 \fi

123 \fi}

Initialise

124 \KV@toks{}

Do the parse

125 \KV@do#1,\relax,

Evaluate the token register outside the group.

126 \expandafter

127 \endgroup

128 \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.

```
129 \let:NN \SetInternalCounter \calc_int_set:Nn
130 \let: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.

```
131 \def\UndeclareRobustCommand#1{%
132   \cs_gundefine:N #1
133   \exp_args:Nc \cs_gundefine:N {\cs_to_str:N #1~}
134 }
135 </package>
```