Copyright 1999 Bjoern Pedersen This program can be redistributed and/or modified under the terms of the LaTeX Project Public License Distributed from CTAN archives in directory macros/latex/base/lppl.txt; either version 1 of the License, or any later version.

This is heavily based on: textmerg.dtx (with options: 'textmerg') Copyright (C) 1992,1994 Mike Piff, University of Sheffield, England

## 1 Intro

This package tries do deal with single character delimited table files. It was mainly inspired and is heavily based on Mike Piff's textmerg package.

Note for package writers: As this package is still under development, the interface is not g guaranteed to be stable. Please consider this if you want to use this package in your style files

# 2 Description

\SetDel The \SetDel<charactertoken> set the character used as a delimiter in the input file. The default is ~. If the character does not have \catcode=12, you should adjust this before setting it as in this example:

> {\catcode'\^^I=12 \SetDel^^I}

This would set del to the *<*tab>-character

**\Fields** The **\Fields** macor takes a list of control sequence, which will be assigned during the read in process. Example:

\Fields{\Title\Forenames\Surname\Address\Grade}

\DelimRead After defining the \Fields, a file is read in with \DelimRead{File}{Template}, where File is the filename of the data file, and Template is the text, in which occurences of the csnames in the \Fields-macro should be replaced by text from the data file. Code

### 2.1 Header

Announce the file.

- 1 (**\*delimtxt**)
- $2 \det 1.02$
- 3 \def\filedate{1999/05/03}
- 4 \def\filename{delimtxt.dtx}
- 5 \def\Copyright{Copyright 1999 Bjoern Pedersen}
- 6 \NeedsTeXFormat{LaTeX2e}[1998/06/01]
- 7 \ProvidesPackage{delimtxt}[\filedate]
- 8 \typeout{Package 'delimtxt' <\filedate>.}
- 9 \typeout{\Copyright}

### 2.2 Utility macros

```
10 \def\glet{\global\let}
11
```

## 2.3 File Handling

This opens a file and reads it line by line into \InputBuffer.

```
12 \newread\DelimFile
13
14 \def\InputFile#1{%
     \openin\DelimFile=#1
15
     \ifeof\DelimFile
16
     \errmessage{Empty Delim file}%
17
     \closein\DelimFile
18
     \long\def\MakeTemplate##1{%
19
20
         \def\Template{}}%
21
     \else\GetInput
22
     \fi}
```

Adjust the catcode of the delimiter temporarily, and read one line of input.

```
23 \def\GetInput{{
24 \global\LF@false
25 \endlinechar=-1%
26 \expandafter\catcode\expandafter`\the\Del=12
27 \global\read\DelimFile to\InputBuffer}}
```

Check, if there is anything left in the Input file. If not, stop Iterating. Empty lines in the file are silently skipped.

```
28 \def\SeeIfEof{%
```

```
\let\NextLook\relax
29
     \ifeof\DelimFile
30
     \else
31
         \ifx\InputBuffer\empty
32
33
            \LookAgain
34
         \fi
     \fi
35
36
     \NextLook}
37
38 \def\LookAgain{\GetInput
     \let\NextLook\SeeIfEof}
39
```

\ifNonBlank \AllowBlank \DontAllowBlank

nBlank We can now prepare to read actual fields from the merge file. A conditional is wBlank used to indicate whether or not the field we are about to read is allowed to be wBlank blank. We also set up a mechanism for changing its value.

40 \newif\ifNonBlank \NonBlankfalse
41 \def\AllowBlank{\global\NonBlankfalse}
42 \def\DontAllowBlank{\global\NonBlanktrue}

## 2.4 Parsing the Input Buffer

This is the difficult part of the processing.

#### 2.4.1 Helper macros and registers

We need some token registers to save the Input, the delimiter, and some of the definitions for dynamic parameter lists

43 \newif\ifLF@

44 \def\mark{\relax}

45 \newtoks\InPutField

- 47 \newtoks\StripT@k
- 48 \newtoks\NextFieldT@k

#### 2.4.2 Strip mark helper

A helper macro to strip of a marker, we placed in the input stream. It is put in \StripT@k and the actual definiton will take place on execution of \SetDel as we need to know what the delimiter actually is.

```
49 \StripT@k={%
50 {%
51 \aftergroup\gdef%
52 \aftergroup\StripMark%
53 \aftergroup#\aftergroup1%
54 \expandafter\aftergroup\the\Del%
55 \aftergroup\mark%
56 }{\gdef\InputBuffer{#1}}
```

#### 2.4.3 Get the next Field value from the input stream

this On execution of \SetDel will define macro mess a This will perform \GetNextInputField#1<expanded \del>#2\lineend. somthing similar to the C language strtck function. This macro gets the contents of the \InputBuffer plus an extra delimiter, a mark and a lineend marker. On Exit \InputBuffer is reassigned with one less Field after stripping of all markers. If nothing is left, a flag is set and \InputBuffer is set empty. This flag is currently unused, but could be used for better error handling in case of missing fields in the input.

```
57
58 \NextFieldT@k={%
59 {%begin of aftergroup group
60 \aftergroup\gdef%
61 \aftergroup\GetNextInputField%
62 \aftergroup#\aftergroup1%
63 \expandafter\aftergroup\the\Del%
64 \aftergroup#\aftergroup2%
65 \aftergroup \lineend}%end of aftergroup group
66 {%
   \if\mark #2%
67
      \global\LF@true%
68
       \glet\InputBuffer=\empty%
69
70
    \else%
       \global\LF@false%
71
       \StripMark#2%
72
```

```
73 \fi%
```

```
\InPutField={#1}%
74
    \if!#1!% check if Field is empty (Ref: D.Carlise in comp.text.tex)
75
76
        \ifNonBlank%
            \MissingField%
77
            \InPutField={???}%
78
        \else%
79
            \InPutField={#1}%
80
81
         \fi%
82
        \else
         \relax
83
    \fi%
84
85 }}
```

This macro sets the Delimiter. As this may be called at any time, we need to redefine the macros \GetNextInputField and \StripMark. The definitions have been stored in two token registers, so we have just to execute them. The trickery with \aftergroup in the token list enables expansion of \the\Del in the macro parameter list.

```
86 \def\SetDel#1{\global\Del={#1}%
87 \the\StripT@k%
88 \the\NextFieldT@k%
89 }
90 \SetDel|
91
92
```

### 2.5 Parsing the fields

Here we parse the inout fields as in the textmerg package, but getting values from our new parser. Probably, the treatment of missing items is not very good( in fact it is completly missing) We have to put a \mark and \lineend in the stream, do detect the end of the input line.

```
93
94 \def\ReadIn#1{%
95 \expandafter\expandafter\expandafter\expandafter\
96 \GetNextInputField%
97 \expandafter\InputBuffer\the\Del%
98 \mark\lineend%
99 \global\edef#1{\the\InPutField}%
100 }
This is not used ust
```

This is not used yet.

```
101 \def\MissingField{%
102 \message{Missing field in file}}
```

Here begins the field parsing, as in the textmerg-package.

```
103
104 \newtoks\GlobalFields
105 %
106 \def\Fields#1{\GlobalFields{#1}}
107 %
108 \def\ParseFields#1{%
109 \ifx#1\EndParseFields%
110 \let\NextParse\relax%
```

```
\ifLF@%
111
           \message{ Line was OK}%
112
113
         \else%
114
           {\message{ There were more items than fields on line
115
           \the\Iteratecounter. They will be skipped.}
116
           \glet\InputBuffer=\empty}%
         \fi%
117
     \else%
118
         \let\NextParse\ParseFields%
119
         \ifx#1+\DontAllowBlank%
120
         \else%
121
            \ifx#1-\AllowBlank%
122
            \else\ReadIn#1%
123
            \fi%
124
         \fi%
125
     \fi\NextParse}%
126
127
128 \let\EndParseFields\ParseFields%
129 \def\ReadFields#1{
130 \ifeof\DelimFile%
131
    \else%
132 \expandafter\ParseFields%
133 \the#1\EndParseFields%
134 \fi}%
```

### 2.6 The iteration code

```
135 \long\def\DelimRead#1#2{\begingroup%
      \InputFile{#1}%
136
      \def\Fields##1{%
137
         \ParseFields##1\EndParseFields}%
138
      \MakeTemplate{#2}\Iterate}%
139
140 \long\def\MakeTemplate#1{\def\Template{#1}}
141 \countdef \Iteratecounter=1%
142
143 \Iteratecounter=0
144 \def\Iterate{%
145
      \global\advance\Iteratecounter by1%
146
      \ReadFields\GlobalFields%
      \Template%
147
      \SeeIfEof%
148
      \ifeof\DelimFile%
149
150
         \def\NextIteration{%
             \endgroup\closein\DelimFile}%
151
152
      \else%
153
         \let\NextIteration\Iterate%
154
      \fi%
      \NextIteration}
155
156 \endinput
```

</delimtxt>

3