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.

```
⟨*delimtxt⟩
\def\fileversion{1.02}
\def\filedate{1999/05/03}
\def\filename{delimtxt.dtx}
\def\Copyright{Copyright 1999 Bjoern Pedersen}
\NeedsTeXFormat{LaTeX2e}[1998/06/01]
\ProvidesPackage{delimtxt}[\filedate]
\typeout{Package `delimtxt´ <\filedate>.}
\typeout{\Copyright}
```

2.2 Utility macros

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

2.3 File Handling

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

```
\newread\DelimFile

\def\InputFile#1{%
   \openin\DelimFile=#1
   \ifeof\DelimFile
   \errmessage{Empty Delim file}%
   \closein\DelimFile
   \long\def\MakeTemplate##1{%
      \def\Template{}}%
   \else\GetInput
   \fi}
```

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

```
\def\GetInput{{
 \global\LF@false
 \endlinechar=-1%
 \expandafter\catcode\expandafter`\the\Del=12
   \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.

```
\def\SeeIfEof{%
   \let\NextLook\relax
   \ifeof\DelimFile
   \else
      \ifx\InputBuffer\empty
         \LookAgain
      \fi
   \fi
   \NextLook}

\def\LookAgain{\GetInput
   \let\NextLook\SeeIfEof}
```

`\ifNonBlank` `\AllowBlank` `\DontAllowBlank`

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

```
\newif\ifNonBlank \NonBlankfalse
\def\AllowBlank{\global\NonBlankfalse}
\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

```
\newif\ifLF@
\def\mark{\relax}
\newtoks\InPutField
\newtoks\Del
\newtoks\StripT@k
\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.

```
\StripT@k={%
{%
\aftergroup\gdef%
\aftergroup\StripMark%
\aftergroup#\aftergroup1%
\expandafter\aftergroup\the\Del%
\aftergroup\mark%
}{\gdef\InputBuffer{#1}}}
```

2.4.3 Get the next Field value from the input stream

On execution of `\SetDel` this mess will define a macro `\GetNextInputField#1<expanded \del>#2\lineend`. This will perform somthing similar to the C language `strtok` 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.

```

\NextFieldT@k={%
{%begin of aftergroup group
\aftergroup\gdef%
\aftergroup\GetNextInputField%
\aftergroup#\aftergroup1%
\expandafter\aftergroup\the\Del%
\aftergroup#\aftergroup2%
\aftergroup\lineend}%end of aftergroup group
{%
 \if\mark #2%
    \global\LF@true%
     \glet\InputBuffer=\empty%
  \else%
     \global\LF@false%
     \StripMark#2%
  \fi%
```

```
  \InPutField={#1}%
  \if!#1!% check if Field is empty (Ref: D.Carlise in comp.text.tex)
      \ifNonBlank%
         \MissingField%
         \InPutField={???}%
      \else%
         \InPutField={#1}%
      \fi%
      \else
      \relax
  \fi%
}}
```

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.

```
\def\SetDel#1{\global\Del={#1}%
\the\StripT@k%
\the\NextFieldT@k%
}
\SetDel|
```

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.

```
\def\ReadIn#1{%
     \expandafter\expandafter\expandafter%
     \GetNextInputField%
     \expandafter\InputBuffer\the\Del%
     \mark\lineend%
      \global\edef#1{\the\InPutField}%
}
```

This is not used yet.

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

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

```
\newtoks\GlobalFields
%
\def\Fields#1{\GlobalFields{#1}}
%
\def\ParseFields#1{%
  \ifx#1\EndParseFields%
  \let\NextParse\relax%
```

```
      \ifLF@%
        \message{ Line was OK}%
      \else%
        {\message{ There were more items than fields on line
        \the\Iteratecounter.  They will be skipped.}
        \glet\InputBuffer=\empty}%
      \fi%
  \else%
      \let\NextParse\ParseFields%
      \ifx#1+\DontAllowBlank%
      \else%
         \ifx#1-\AllowBlank%
         \else\ReadIn#1%
         \fi%
      \fi%
  \fi\NextParse}%

\let\EndParseFields\ParseFields%

\def\ReadFields#1{
\ifeof\DelimFile%
  \else%
\expandafter\ParseFields%
  \the#1\EndParseFields%
\fi}%
```

2.6 The iteration code

```
\long\def\DelimRead#1#2{\begingroup%
   \InputFile{#1}%
   \def\Fields##1{%
      \ParseFields##1\EndParseFields}%
   \MakeTemplate{#2}\Iterate}%

\long\def\MakeTemplate#1{\def\Template{#1}}

\countdef\Iteratecounter=1%

\Iteratecounter=0
\def\Iterate{%
   \global\advance\Iteratecounter by1%
   \ReadFields\GlobalFields%
   \Template%
   \SeeIfEof%
   \ifeof\DelimFile%
      \def\NextIteration{%
         \endgroup\closein\DelimFile}%
   \else%
      \let\NextIteration\Iterate%
   \fi%
   \NextIteration}
\endinput
```

</delimtxt>

3