The **parselines*** package

a simple line parser for TeX

FC

2011/02/19 – version 1.4

Abstract

parselines provides an environment “`parse lines`” which applies a macro to each line of the input between `\begin {parse lines}` and `\end {parse lines}`.

There is also a macro: `\dofilebyline`: its purpose is to expand a user-defined macro `\do` with one parameter (the line) on each line of a given text file.

This packages requires ε-TEXand no other package.

Contents

1 User interface

1.1 The `parse lines` environment

```
\begin {⟨parse lines⟩}[initialisation]{⟨replacement text of a macro with 1 parameter⟩}
\end {⟨parse lines⟩}
```

Example (with xcolor) – inside the environment `\row` expands to the current row number (stored in plain TEX register `\count@`):

```
\definecolorseries{serie}{rgb}{last}{Goldenrod}{DarkSeaGreen}
\resetcolorseries[3]{serie}
\begin{parse lines}{\color{serie!!+}‘‘#1’’\par}
   This is a nice little package
   and I really don’t know
   What to do with it !
   May be tomorrow...
\end{parse lines}
```

“This is a nice little package”

“and I really don’t know”

“What to do with it !”

“May be tomorrow...”

This documentation is produced with the +DocStrip+ utility.

⟶ To get the documentation, run (thrice): `pdflatex parselines.dtx`
To get the index, run: `makeindex -s gind.ist parselines.idx`
⟶ To get the package, run: `etex parselines.dtx`

The `.dtx` file is embedded into this pdf file thank to embedfile by H. Oberdiek.

```
\begin{parse lines}{\row: #1\par}
   This is a nice little package
   and I really don't know
   What to do with it !
   May be tomorrow...
\end{parse lines}
```

1: This is a nice little package

2: and I really don't know

3: What to do with it !

4: May be tomorrow...

1.2 `\dofilebyline`

`\dofilebyline {⟨file-name⟩}`
`\dofilebyline * {⟨file-name⟩}`

Expands the macro **`\do`** on each line of a given file.

Each line is read using the TEX primitive **`\read`**. By default, `\endlinechar` is suppressed (`\endlinechar =-1`) in order not to add a space at the end of each line read. If you do not want this behaviour, use the ∗ star form.

If the file does not exist, an error is displayed.

Example:

```
\def\do#1{\ifx\par#1\else \addto@hook\mytoken{#1}\fi}%
\dofilebyline{file name.tex}
```

In this example: `\ifx \par#1\else` will evaluate to `false` if `#1` is empty. Therefore, empty lines and double empty lines (equivalent to `\par`) are not captured into the token `\mytoken`.

2 Implementation

2.1 Identification

The package namespace is **`parselin@`**.

```
⟨*package⟩
\NeedsTeXFormat{LaTeX2e}% LaTeX 2.09 can't be used (nor non-LaTeX)
   [2005/12/01]% LaTeX must be 2005/12/01 or younger
\ProvidesPackage{parselines}
         [2011/02/19 v1.4 - a simple line parser for TeX]
```

2.2 Collecting stuff inside an environment

```
\ifdefined\globtoks  \globtoks\parselin@tk
\else                \newtoks\parselin@tk
\fi
\newcommand\CollectEnvir{}
\protected\def\CollectEnvir#1{%
   \gdef\parselin@Finish{#1%
      \expandafter{\the\parselin@tk}%
      \global\let\parselin@Finish\@undefined}%
   \ifx\parselin@tk#1\else
      \expandafter\def\expandafter\parselin@Finish
         \expandafter{\parselin@Finish\parselin@tk{}}%
      \if \relax
          \expandafter\expandafter\expandafter\parselin@isatoken
            \expandafter\meaning\expandafter#1\string\toks
          \relax
         \expandafter\gdef\expandafter\parselin@Finish
            \expandafter{\expandafter\expandafter
               \expandafter\def\expandafter\expandafter
               \parselin@Finish}%
   \fi\fi
   \parselin@tk{}\def\parselin@stack{b}\parselin@Collect@Body
}% \CollectEnvir
\long\def\parselin@Collect@Body#1\end#2{%
  \edef\parselin@stack{\parselin@PushBegins#1\begin\end
                     \expandafter\@gobble\parselin@stack}%
  \ifx\parselin@stack\@empty
      \global\parselin@tk\expandafter{\the\parselin@tk#1}%
      \aftergroup\parselin@Finish
      \end{#2}%
  \else
      \parselin@tk\expandafter{\the\parselin@tk#1\end#2}%
      \expandafter\parselin@Collect@Body % recurse
  \fi
}% \parselin@Collect@Body
\long\def\parselin@PushBegins#1\begin#2{%
      \ifx\end#2\else b\expandafter\parselin@PushBegins\fi}
\expandafter\def\expandafter\parselin@isatoken
   \expandafter#\expandafter1\string\toks#2\relax{\detokenize{#2}\relax}%
```

2.3 The `parse lines` environment

```
\newenvironment{parse lines}[2][]
{%
   #1%
   \count@\z@\def\row{\number\count@}%
   \def\parselin@ProcessLine##1{\advance\count@\@ne #2}%
   \parselin@endlinechar{%
      \def~##1~{%
         \ifx\parselin@@@@@@##1\relax
         \else \parselin@ProcessLine{##1}\expandafter~%
         \fi}%
   }%
   \catcode\endlinechar\active
   \CollectEnvir\parselin@tk
}
{%
   \parselin@endlinechar{%
      \parselin@tk\expandafter{\the\parselin@tk\parselin@@@@@@~}%
   }\the\parselin@tk
}%
\long\def\parselin@endlinechar#1{\begingroup
   \lccode`\~\endlinechar  \lowercase{\endgroup#1}%
}% \parselin@endlinechar
\def\parselin@@@@@@{\parselin@@@@@@}
```

2.4 `\dofilebyline`

```
\newread\parselin@read
\protected\def\dofilebyline{\let\parselin@Finish\@empty
   \@ifstar
      \parselin@dobyline
      {\edef\parselin@Finish{\endlinechar\the\endlinechar\relax}%
       \endlinechar\m@ne
       \parselin@dobyline}%
}% \dofilebyline
\protected\def\parselin@dobyline#1{% #1 = file name
   \IfFileExists{#1}
77       \parselin@do@byline
78       {\PackageError{parselines}
79          {No file #1 found !
80          \MessageBreak Your command \string\dobyline\space was ignored}\@eha
81       }%
82 }% \parselin@dobyline
83 \def\parselin@do@byline{%
84    \openin\parselin@read=\@filef@und%
85    \loop
86       \ifeof\parselin@read\else
87          \read\parselin@read to\@tempa
88          \expandafter\do\expandafter{\@tempa}%
89    \repeat
90    \parselin@Finish
91 }% \parselin@do@byline
```

92 ⟨/package⟩

3 History

[2011/02/19 v1.4]

- Recompilation of the documentation to conform to tabu[1] v2.5 and interfaces[2] v3.1

[2010/11/20 v1.3]

- Replacement of `\newrobustcmd` by `\protected` `\def`: parselines does not require etoolbox...

[2010/11/16 v1.2]

- Fix in CTAN archive (`.ins` file).

[2010/10/23 v1.1]

- `\row` counter added
- Improved documentation.

[2010/06/20 v1.0]

- First version.

4 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; plain numbers refer to the code lines where the entry is used.

---

[1] tabu: CTAN:macros/latex/contrib/tabu

[2] interfaces: CTAN:macros/latex/contrib/interfaces