The enparen package

Heiko Oberdiek*
<heiko.oberdiek at googlemail.com>

2016/05/16 v1.1

Abstract

The package defines macros to set parentheses that automatically change the symbols from inner to outer fences.

Contents

1 Documentation **2**
1.1 User macros . . . 2
1.2 Contexts . . . 3
1.3 Options . . . 3
1.4 Notes . . . 4

2 Implementation **4**
2.1 Resources . . . 4
2.2 Contexts . . . 4
2.2.1 Stack for contexts . . . 5
2.2.2 Context user macros . . . 5
2.3 Symbols . . . 6
2.4 Main user macros . . . 6
2.5 Options . . . 8
2.6 Context settings . . . 8
2.7 At end of document . . . 8

3 Installation **9**
3.1 Download . . . 9
3.2 Bundle installation . . . 9
3.3 Package installation . . . 9
3.4 Refresh file name databases . . . 10
3.5 Some details for the interested . . . 10

4 References **10**

5 History **10**
[2012/01/07 v1.0] . . . 10
[2016/05/16 v1.1] . . . 10

6 Index **11**

*Please report any issues at https://github.com/ho-tex/oberdiek/issues

1 Documentation

The LaTeX package provides macros for automatically changed parentheses symbols depending on the fence order. The innermost parentheses are always using the same parentheses. The symbols changes for the outer fences.

Example:

{a (b) [c (d)] (e)}

is generated by

```
\documentclass{article}
\usepackage{enparen}
\begin{document}
 \enparen{a \enparen{b} \enparen{c \enparen{d}} \enparen{e}}
\end{document}
```

The package is the result of a newsgroup thread. Dan gives the following specification [1]:

> "On the other hand, the rules for fences are usually the reverse: innermost fences are always `()`, next outer are `[]`, etc. This means the opening fence has to wait until all the fences between it and the matching close have been detected before it can decide whether to be `(` or `[` or `{`."

The fence level counting starts from innermost parentheses with one. For the next outer fences the level is increased by one. The example above with level indexes:

$\{_3\mathrm{a}\ (_1\mathrm{b})_1\ [_2\mathrm{c}\ (_1\mathrm{d})_1]_2\ (\mathrm{e})\}_3$

The correct level is only known at the closing symbol. Therefore the correct value is remembered in the main `.aux` file and used in the second LaTeX run.

1.1 User macros

`\enparen {`⟨*text*⟩`}`

The macro `\enparen` puts its argument ⟨*text*⟩ in parentheses. If the macro is nested, the used fence symbols change for the outer fences.

`\enparenLeft`
`\enparenRight`

Instead of `\enparen{text}` the left and right symbol can be used separately, but in pairs:

```
\enparenLeft text\enparenRight
```

`\enparenLeft` and `\enparenRight` may be used at different group levels, but they must be properly nested.

`\enparenSetSymbols {`⟨*level*⟩`} {`⟨*opening symbol*⟩`} {`⟨*closing symbol*⟩`}`

Macro `\enparenSetSymbols` configures the ⟨*opening symbol*⟩ and ⟨*closing symbol*⟩ for the ⟨*level*⟩. The ⟨*level*⟩ is a number and the counting starts with one. Level zero is used, if the correct level is not known (e.g. in the first LaTeX run). The package defines the following sets:

```
\enparenSetSymbols{0}{(}{)}
\enparenSetSymbols{1}{(}{)}
\enparenSetSymbols{2}{[}{]}
\enparenSetSymbols{3}{\{}{\}}
```

Example for changing the third and adding a fourth level:

```
\enparenSetSymbols{3}{\ensuremath{\langle}}{\ensuremath{\rangle}}
\enparenSetSymbols{4}{\{}{\}}
```

`\enparenUnsetSymbols {`⟨*level*⟩`}`

The symbols for level ⟨*level*⟩ are removed. Example scenario: Only two nesting levels must be used, the package defines more, then the third level can be disabled by `\enparenUnsetSymbols{3}` and the user gets warnings if parentheses at level 3 are needed.

1.2 Contexts

`\enparenBeginContext {`⟨*name*⟩`}`
`\enparenEndContext {`⟨*name*⟩`}`

If the current text is interrupted by footnotes, floats with captions, then the parentheses inside the text of footnotes, captions, …should restart from scratch. This can be achieved by embedding the text inside macros `\enparenBeginContext` and `\enparenEndContext`. These macros must be properly nested. The ⟨*name*⟩ for the begin and end macro must be the same. It is a help for debugging problems, because the warning messages show the context name. But it is not necessary that the begin/end pairs have different names. Example:

```
\enparenLeft text before table ...
\begin{table}
  \caption{Table caption}
  \enparenBegin{Context}{table}
  Other text \enparen{foobar}.
  \enparenEnd{Context}{table}
\end{table}
text after table ...
\enparenRight
```

The parentheses inside the table environment and context ‘table’ are not nested inside other parentheses: (foobar). In case of captions and footnotes the contexts are automatically added, see next section about options.

1.3 Options

`\enparenSetup {`⟨*key value list*⟩`}`

Some options (currently all) can also be set after the package is loaded. They can be set in the argument ⟨*key value list*⟩ of `\enparenSetup`. Options are disabled after they are used the last time. Currently all options are boolean options and are disabled in `\begin{document}`.

caption: The caption text is put in a context `caption`.

footnote: The footnote text is put in a context `footnote`.

Example for disabling the two options at different places:

```
\usepackage[caption=false]{enparen}
\enparenSetup{footnote=false}
```

1.4 Notes

Implicite kerning: Unexpandable stuff might affect the implicite kerning. The package cannot avoid this, because it need to define and redefine macros at the occurence of each symbol. This is done before the opening and after the closing symbol, thus that the implicite kerning inside is not affected.

2 Implementation

```
1 ⟨*package⟩
2 \NeedsTeXFormat{LaTeX2e}
3 \ProvidesPackage{enparen}
4   [2016/05/16 v1.1 Parentheses nesting (HO)]%
```

2.1 Resources

```
5 \RequirePackage{ltxcmds}[2011/11/09]
6 \ltx@IfUndefined{numexpr}{%
7   \PackageError{enparen}{%
8     Missing e-TeX's \ltx@backslashchar numexpr.\MessageBreak
9     The package will continue with emergency definitions%
10  }\@ehc
11  \def\enparenLeft{(}%
12  \def\enparenRight{)}%
13  \long\def\enparen#1{\enparenLeft#1\enparenRight}%
14  \let\enparenSetup\ltx@gobble
15  \let\enparenSetSymbols\ltx@gobblethree
16  \let\enparenUnsetSymbols\ltx@gobble
17  \endinput
18 }{}
19 \RequirePackage{protecteddef}[2011/01/31]
20 \RequirePackage{atveryend}[2011/06/30]
21 \RequirePackage{uniquecounter}[2011/01/30]
22 \RequirePackage{zref-base}[2011/03/18]
23 \RequirePackage{kvoptions}[2011/06/30]
24 \RequirePackage{kvsetkeys}[2011/10/18]
```

\zref@wrapper@mainaux

```
25 \providecommand{\zref@wrapper@mainaux}[1]{%
26   \ifx\@auxout\@mainaux
27     #1%
28   \else
29     \begingroup
30       \let\@auxout\@mainaux
31       #1%
32     \endgroup
33   \fi
34 }
35 \UniqueCounterNew{enparen}
36 \zref@newprop{enparen}[0]{}
```

2.2 Contexts

\enparenContextDefault

```
37 \def\enparenContextDefault{main}
```

\enparen@ctx

```
38 \let\enparen@ctx\ltx@empty
```

\enparen@stack

```
39 \let\enparen@stack\ltx@empty
```

2.2.1 Stack for contexts

\enparen@CtxStack

```
\global\let\enparen@CtxStack\ltx@empty
```

\enparen@CtxStackPush

```
\def\enparen@CtxStackPush#1{%
  \xdef\enparen@CtxStack{%
    {\enparen@ctx}{\enparen@stack}%
    \enparen@CtxStack
  }%
  \xdef\enparen@ctx{#1}%
  \global\let\enparen@stack\ltx@empty
}
```

\enparen@CtxStackPop

```
\def\enparen@CtxStackPop{%
  \ifx\enparen@CtxStack\ltx@empty
    \PackageWarning{enparen}{%
      Pop request for empty context stack%
    }%
    \global\let\enparen@ctx\enparenContextDefault
    \global\let\enparen@stack\ltx@empty
  \else
    \xdef\enparen@ctx{%
      \expandafter\ltx@car\enparen@CtxStack\@nil
    }%
    \xdef\enparen@stack{%
      \expandafter\ltx@carsecond\enparen@CtxStack\@nil
    }%
    \xdef\enparen@CtxStack{%
      \expandafter\ltx@cdrtwo\enparen@CtxStack\@nil
    }%
  \fi
}
```

2.2.2 Context user macros

\enparenBeginContext

```
\ProtectedDef*{\enparenBeginContext}[1]{%
  \enparen@CtxStackPush{#1}%
}
```

\enparenEndContext

```
\ProtectedDef*{\enparenEndContext}[1]{%
  \edef\enparen@temp{#1}%
  \ifx\enparen@temp\enparen@ctx
  \else
    \PackageWarning{enparen}{%
      Context mismatch in end request.\MessageBreak
      '#1' should be ended, but current context\MessageBreak
      is '\enparen@ctx'%
    }%
  \fi
  \enparenCheckEmptyStack
  \enparen@CtxStackPop
}
```

\enparenCheckEmptyStack

```
\ProtectedDef*{\enparenCheckEmptyStack}[0]{%
  \ifx\enparen@stack\ltx@empty
  \else
```

```
    \PackageWarning{enparen}{%
      Ending non-empty stack '\enparen@ctx':\MessageBreak
      \enparen@PrintStack\MessageBreak
    }%
  \fi
}
```

\enparen@PrintStack

```
\def\enparen@PrintStack{%
  \expandafter\enparen@PrintStackAux
  \enparen@stack\ltx@empty\ltx@empty
}
```

\enparen@PrintStackAux

```
\def\enparen@PrintStackAux#1#2{%
  \ifx\ltx@empty#1%
  \else
    {#1:#2}%
    \expandafter\enparen@PrintStackAux
  \fi
}
```

2.3 Symbols

\enparenSetSymbols

```
\ProtectedDef*{\enparenSetSymbols}[3]{%
  \expandafter
  \def\csname enparen@symbol\the\numexpr#1L\endcsname{#2}%
  \expandafter
  \def\csname enparen@symbol\the\numexpr#1R\endcsname{#3}%
}
```

\enparenUnsetSymbols

```
\ProtectedDef*{\enparenUnsetSymbols}[1]{%
  \expandafter
  \let\csname enparen@symbol\the\numexpr#1L\endcsname\ltx@undefined
  \expandafter
  \let\csname enparen@symbol\the\numexpr#1R\endcsname\ltx@undefined
}
```

```
\enparenSetSymbols{0}{(}{)}
\enparenSetSymbols{1}{(}{)}
\enparenSetSymbols{2}{[}{]}
\enparenSetSymbols{3}{\{}{\}}
\enparenSetSymbols{4}{\ensuremath{\langle}}{\ensuremath{\rangle}}
```

2.4 Main user macros

\enparen

```
\ProtectedDef{\enparen}[1]{%
  \enparenLeft#1\enparenRight
}
```

\enparenLeft

```
\ProtectedDef*{\enparenLeft}[0]{%
  \UniqueCounterCall{enparen}\enparen@Left
}
```

\enparen@Left

```
\def\enparen@Left#1{%
  \xdef\enparen@stack{%
```

```
    {#1}{1}%
    \expandafter\enparen@Inc\expandafter2\expandafter!%
    \enparen@stack\ltx@empty\ltx@empty
  }%
  \edef\enparen@tmp{\zref@extract{enparen#1}{enparen}}%
  \ltx@IfUndefined{enparen@symbol\enparen@tmp L}{%
    \PackageWarning{enparen}{%
      Undefined symbols for level \enparen@tmp
    }%
    \csname enparen@symbol0L\endcsname
  }{%
    \csname enparen@symbol\enparen@tmp L\endcsname
  }%
}
```

\enparen@Inc

```
\def\enparen@Inc#1!#2#3{%
  \ifx\ltx@empty#2%
  \else
    \ifnum#3<#1 %
      {#2}{#1}%
      \expandafter\enparen@Inc
      \the\numexpr#1+1\expandafter\expandafter\expandafter!%
    \else
      {#2}{#3}%
    \fi
  \fi
}
```

\enparenRight

```
\ProtectedDef*{\enparenRight}[0]{%
  \ifx\enparen@stack\ltx@empty
    \PackageWarning{enparen}{%
      Missing left symbol for right symbol%
    }%
    \csname enparen@symbol0R\endcsname
  \else
    \expandafter\enparen@Right\enparen@stack\@nil
  \fi
}
```

\enparen@Right

```
\def\enparen@Right#1#2#3\@nil{%
  \ltx@IfUndefined{%
    enparen@symbol%
    \zref@extract{enparen#1}{enparen}%
    R%
  }{%
    \csname enparen@symbol0R\endcsname
  }{%
    \csname
      enparen@symbol%
      \zref@extract{enparen#1}{enparen}%
       R%
    \endcsname
  }%
  \zref@wrapper@mainaux{%
    \zref@setcurrent{enparen}{#2}%
    \zref@labelbyprops{enparen#1}{enparen}%
  }%
  \xdef\enparen@stack{#3}%
}
```

2.5 Options

```
\SetupKeyvalOptions{%
  family=enparen,%
  prefix=enparen@,%
}
```

\enparenSetup

```
\ProtectedDef*{\enparenSetup}[0]{%
  \kvsetkeys{enparen}%
}

\DeclareBoolOption[true]{footnote}
\DeclareBoolOption[true]{caption}
\ProcessKeyvalOptions*
```

2.6 Context settings

\enparen@AtBegin

```
\def\enparen@AtBegin{%
  \ifenparen@footnote
    \let\enparen@org@makefntext\@makefntext
    \long\def\@makefntext##1{%
      \enparen@org@makefntext{%
        \enparenBeginContext{footnote}%
        ##1%
        \enparenEndContext{footnote}%
      }%
    }%
  \fi
  \enparen@Disable{footnote}%
  \ifenparen@caption
    \let\enparen@org@makecaption\@makecaption
    \long\def\@makecaption##1##2{%
      \enparen@org@makecaption{##1}{%
        \enparenBeginContext{caption}%
        ##2%
        \enparenEndContext{caption}%
      }%
    }%
  \fi
  \enparen@Disable{caption}%
}
```

\enparen@Disable

```
\def\enparen@Disable#1{%
  \DisableKeyvalOption[%
    action=warning,%
    package=enparen,%
  ]{enparen}{#1}%
}

\AtBeginDocument{\enparen@AtBegin}
```

2.7 At end of document

\enparen@AtEnd

```
\def\enparen@AtEnd{%
  \enparenCheckEmptyStack
  \ifx\enparen@CtxStack\ltx@empty
  \else
    \PackageWarningNoLine{enparen}{%
      Context stack is not empty at end of document.\MessageBreak
```

```
      Current stack and contents of context stack:\MessageBreak
      [\enparen@ctx]:[\enparen@PrintStack]%
      \expandafter
      \enparen@PrintContextStack\enparen@CtxStack\relax\relax
    }%
  \fi
}
```

\enparen@PrintContextStack

```
\def\enparen@PrintContextStack#1#2{%
  \ifx\relax#1\ltx@empty
  \else
    \MessageBreak
    [#1]:[\enparen@PrintStackAux#2\ltx@empty\ltx@empty]% hash-ok
    \expandafter\enparen@PrintContextStack
  \fi
}
\AtVeryEndDocument{\enparen@AtEnd}
⟨/package⟩
```

3 Installation

3.1 Download

Package. This package is available on CTAN[1]:

CTAN:macros/latex/contrib/oberdiek/enparen.dtx The source file.

CTAN:macros/latex/contrib/oberdiek/enparen.pdf Documentation.

Bundle. All the packages of the bundle 'oberdiek' are also available in a TDS compliant ZIP archive. There the packages are already unpacked and the documentation files are generated. The files and directories obey the TDS standard.

CTAN:install/macros/latex/contrib/oberdiek.tds.zip

TDS refers to the standard "A Directory Structure for TEX Files" (CTAN:tds/tds.pdf). Directories with texmf in their name are usually organized this way.

3.2 Bundle installation

Unpacking. Unpack the oberdiek.tds.zip in the TDS tree (also known as texmf tree) of your choice. Example (linux):

```
unzip oberdiek.tds.zip -d ~/texmf
```

Script installation. Check the directory TDS:scripts/oberdiek/ for scripts that need further installation steps. Package attachfile2 comes with the Perl script pdfatfi.pl that should be installed in such a way that it can be called as pdfatfi. Example (linux):

```
chmod +x scripts/oberdiek/pdfatfi.pl
cp scripts/oberdiek/pdfatfi.pl /usr/local/bin/
```

3.3 Package installation

Unpacking. The .dtx file is a self-extracting docstrip archive. The files are extracted by running the .dtx through plain TEX:

```
tex enparen.dtx
```

[1] http://ctan.org/pkg/enparen

TDS. Now the different files must be moved into the different directories in your installation TDS tree (also known as texmf tree):

```
enparen.sty → tex/latex/oberdiek/enparen.sty
enparen.pdf → doc/latex/oberdiek/enparen.pdf
enparen.dtx → source/latex/oberdiek/enparen.dtx
```

If you have a docstrip.cfg that configures and enables docstrip's TDS installing feature, then some files can already be in the right place, see the documentation of docstrip.

3.4 Refresh file name databases

If your TEX distribution (teTEX, mikTEX, ...) relies on file name databases, you must refresh these. For example, teTEX users run texhash or mktexlsr.

3.5 Some details for the interested

Unpacking with LATEX. The .dtx chooses its action depending on the format:

plain TEX: Run docstrip and extract the files.

LATEX: Generate the documentation.

If you insist on using LATEX for docstrip (really, docstrip does not need LATEX), then inform the autodetect routine about your intention:

```
latex \let\install=y\input{enparen.dtx}
```

Do not forget to quote the argument according to the demands of your shell.

Generating the documentation. You can use both the .dtx or the .drv to generate the documentation. The process can be configured by the configuration file ltxdoc.cfg. For instance, put this line into this file, if you want to have A4 as paper format:

```
\PassOptionsToClass{a4paper}{article}
```

An example follows how to generate the documentation with pdfLATEX:

```
pdflatex enparen.dtx
makeindex -s gind.ist enparen.idx
pdflatex enparen.dtx
makeindex -s gind.ist enparen.idx
pdflatex enparen.dtx
```

4 References

[1] Dan Luecking: *Re: bracket order*; newsgroup comp.text.tex; message id ⟨9b07c9c8-ff92-4cbf-b3a9-84efecfeb506@l24g2000yqm.googlegroups.com⟩ 2012-01-05.
http://groups.google.com/group/comp.text.tex/msg/8774519da31c2352

5 History

[2012/01/07 v1.0]

- First version.

[2016/05/16 v1.1]

- Documentation updates.

6 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.

Symbols

\@auxout 26, 30
\@ehc 10
\@mainaux 26, 30
\@makecaption 208, 209
\@makefntext 197, 198
\@nil 58, 61, 64, 162, 165
\{ 119
\} 119

A

\AtBeginDocument 225
\AtVeryEndDocument 247

C

\csname 106, 108, 112, 114, 138, 140, 160, 171, 173

D

\DeclareBoolOption 192, 193
\DisableKeyvalOption 220

E

\endcsname 106, 108, 112, 114, 138, 140, 160, 171, 177
\endinput 17
\enparen *2*, 13, 121
\enparen@AtBegin 195, 225
\enparen@AtEnd 226, 247
\enparen@ctx 38, 43, 46, 54, 57, 73, 78, 88, 233
\enparen@CtxStack 40, 42, 44, 50, 58, 61, 63, 64, 228, 235
\enparen@CtxStackPop 49, 82
\enparen@CtxStackPush 41, 69
\enparen@Disable 206, 217, 219
\enparen@Inc 130, 143
\enparen@Left 125, 127
\enparen@org@makecaption . . . 208, 210
\enparen@org@makefntext 197, 199
\enparen@PrintContextStack . . 235, 239
\enparen@PrintStack 89, 93, 233
\enparen@PrintStackAux . . . 94, 97, 243
\enparen@Right 162, 165
\enparen@stack 39, 43, 47, 55, 60, 85, 95, 128, 131, 156, 162, 183
\enparen@temp 72, 73
\enparen@tmp 133, 134, 136, 140
\enparenBeginContext . . *3*, 68, 200, 211
\enparenCheckEmptyStack . 81, 84, 227
\enparenContextDefault 37, 54
\enparenEndContext 71, 202, 213
\enparenLeft *2*, 11, 13, 122, 124
\enparenRight 12, 13, 122, 155
\enparenSetSymbols *2*, 15, 104, 116, 117, 118, 119, 120
\enparenSetup *3*, 14, 189
\enparenUnsetSymbols *3*, 16, 110
\ensuremath 120

I

\ifenparen@caption 207
\ifenparen@footnote 196
\ifnum 146
\ifx 26, 50, 73, 85, 98, 144, 156, 228, 240

K

\kvsetkeys 190

L

\langle 120
\ltx@backslashchar 8
\ltx@car 58
\ltx@carsecond 61
\ltx@cdrtwo 64
\ltx@empty 38, 39, 40, 47, 50, 55, 85, 95, 98, 131, 144, 156, 228, 240, 243
\ltx@gobble 14, 16
\ltx@gobblethree 15
\ltx@IfUndefined 6, 134, 166
\ltx@undefined 112, 114

M

\MessageBreak 8, 76, 77, 88, 89, 231, 232, 242

N

\NeedsTeXFormat 2
\numexpr 106, 108, 112, 114, 149

P

\PackageError 7
\PackageWarning . 51, 75, 87, 135, 157
\PackageWarningNoLine 230
\ProcessKeyvalOptions 194
\ProtectedDef 68, 71, 84, 104, 110, 121, 124, 155, 189
\providecommand 25
\ProvidesPackage 3

R

\rangle 120
\RequirePackage 5, 19, 20, 21, 22, 23, 24

S

\SetupKeyvalOptions 185

T

\the 106, 108, 112, 114, 149

U

\UniqueCounterCall 125
\UniqueCounterNew 35

Z

\zref@extract 133, 168, 175
\zref@labelbyprops 181
\zref@newprop 36
\zref@setcurrent 180
\zref@wrapper@mainaux 25, 179