The enparen package

Heiko Oberdiek*

2016/05/16 v1.1

Abstract

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

Contents

*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:

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

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

2.1 Resources

```
\RequirePackage{ltxcmds}[2011/11/09]
\ltx@IfUndefined{numexpr}{%
  \PackageError{enparen}{%
    Missing e-TeX's \ltx@backslashchar numexpr.\MessageBreak
    The package will continue with emergency definitions%
  }\@ehc
  \def\enparenLeft{(}%
  \def\enparenRight{)}%
  \long\def\enparen#1{\enparenLeft#1\enparenRight}%
  \let\enparenSetup\ltx@gobble
  \let\enparenSetSymbols\ltx@gobblethree
  \let\enparenUnsetSymbols\ltx@gobble
  \endinput
}{}
\RequirePackage{protecteddef}[2011/01/31]
\RequirePackage{atveryend}[2011/06/30]
\RequirePackage{uniquecounter}[2011/01/30]
\RequirePackage{zref-base}[2011/03/18]
\RequirePackage{kvoptions}[2011/06/30]
\RequirePackage{kvsetkeys}[2011/10/18]
```

`\zref@wrapper@mainaux`

```
\providecommand{\zref@wrapper@mainaux}[1]{%
  \ifx\@auxout\@mainaux
    #1%
  \else
    \begingroup
      \let\@auxout\@mainaux
      #1%
    \endgroup
  \fi
}
```

```
\UniqueCounterNew{enparen}
\zref@newprop{enparen}[0]{}
```

2.2 Contexts

\enparenContextDefault

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

\enparen@ctx

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

\enparen@stack

```
\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@symbolOR\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:pkg/tds). 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
```

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
```

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 (TEX Live, MiKTEX, ...) relies on file name databases, you must refresh these. For example, TEX Live 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.

[1] CTAN:pkg/enparen

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. https://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.