latexindent.pl is a Perl script that indents .tex (and other) files according to an indentation scheme that the user can modify to suit their taste. Environments, including those with alignment delimiters (such as `tabular`), and commands, including those that can split braces and brackets across lines, are usually handled correctly by the script. Options for `verbatim`-like environments and commands, together with indentation after headings (such as `chapter`, `section`, etc) are also available. The script also has the ability to modify line breaks, and to add comment symbols and blank lines; furthermore, it permits string or regex-based substitutions. All user options are customisable via the switches and the YAML interface.

tl;dr, a quick start guide is given in Section 1.3 on page 5.

### Contents

1 Introduction 5
   1.1 Thanks .......................................................... 5
   1.2 License .......................................................... 5
   1.3 Quick start ..................................................... 5
   1.4 Required perl modules ........................................... 11
   1.5 About this documentation ....................................... 11
   1.6 A word about regular expressions ............................. 12

2 Demonstration: before and after 13

3 How to use the script 14
   3.1 Requirements .................................................. 14
       3.1.1 Perl users ................................................. 14
       3.1.2 Windows users without perl .............................. 14
       3.1.3 Ubuntu Linux users without perl ....................... 14
       3.1.4 macOS users without perl ................................ 14
       3.1.5 conda users .............................................. 14

\*and contributors! See Section 11.5 on page 158. For all communication, please visit [35].
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1.6 docker users</td>
<td>14</td>
</tr>
<tr>
<td>3.2 From the command line</td>
<td>15</td>
</tr>
<tr>
<td>3.3 From arara</td>
<td>21</td>
</tr>
<tr>
<td>3.4 Summary of exit codes</td>
<td>21</td>
</tr>
<tr>
<td>4 indentconfig.yaml, local settings and the -y switch</td>
<td>23</td>
</tr>
<tr>
<td>4.1 indentconfig.yaml and _indentconfig.yaml</td>
<td>23</td>
</tr>
<tr>
<td>4.2 localSettings.yaml and friends</td>
<td>24</td>
</tr>
<tr>
<td>4.3 The -y.yaml switch</td>
<td>25</td>
</tr>
<tr>
<td>4.4 Settings load order</td>
<td>25</td>
</tr>
<tr>
<td>4.5 indentconfig.yaml and .indentconfig.yaml</td>
<td>23</td>
</tr>
<tr>
<td>4.6 localSettings.yaml and friends</td>
<td>24</td>
</tr>
<tr>
<td>4.7 The -y</td>
<td>yaml switch</td>
</tr>
<tr>
<td>4.8 Settings load order</td>
<td>25</td>
</tr>
<tr>
<td>4.9 indentconfig.yaml and _indentconfig.yaml</td>
<td>23</td>
</tr>
<tr>
<td>4.10 localSettings.yaml and friends</td>
<td>24</td>
</tr>
<tr>
<td>4.11 The -y</td>
<td>yaml switch</td>
</tr>
<tr>
<td>4.12 Settings load order</td>
<td>25</td>
</tr>
<tr>
<td>5 defaultSettings.yaml</td>
<td>27</td>
</tr>
<tr>
<td>5.1 Backup and log file preferences</td>
<td>27</td>
</tr>
<tr>
<td>5.2 Verbatim code blocks</td>
<td>29</td>
</tr>
<tr>
<td>5.3 filecontents and preamble</td>
<td>32</td>
</tr>
<tr>
<td>5.4 Indentation and horizontal space</td>
<td>33</td>
</tr>
<tr>
<td>5.5 Aligning at delimiters</td>
<td>34</td>
</tr>
<tr>
<td>5.5.1 lookForAlignDelims: spacesBeforeAmpersand</td>
<td>39</td>
</tr>
<tr>
<td>5.5.2 lookForAlignDelims: alignFinalDoubleBackSlash</td>
<td>40</td>
</tr>
<tr>
<td>5.5.3 lookForAlignDelims: the dontMeasure feature</td>
<td>41</td>
</tr>
<tr>
<td>5.5.4 lookForAlignDelims: the delimiterRegEx and delimiterJustification feature</td>
<td>44</td>
</tr>
<tr>
<td>5.5.5 lookForAlignDelims: lookForChildCodeBlocks</td>
<td>46</td>
</tr>
<tr>
<td>5.5.6 lookForAlignDelims: alignContentAfterDoubleBackSlash</td>
<td>46</td>
</tr>
<tr>
<td>5.5.7 lookForAlignDelims: the delimiterRegEx and delimiterJustification feature</td>
<td>46</td>
</tr>
<tr>
<td>5.5.8 lookForAlignDelims: alignContentAfterDoubleBackSlash</td>
<td>46</td>
</tr>
<tr>
<td>5.5.9 lookForAlignDelims: the delimiterRegEx and delimiterJustification feature</td>
<td>46</td>
</tr>
<tr>
<td>5.5.10 lookForAlignDelims: alignContentAfterDoubleBackSlash</td>
<td>46</td>
</tr>
<tr>
<td>5.6 Indent after items, specials and headings</td>
<td>47</td>
</tr>
<tr>
<td>5.7 The code blocks known latexindent.pl</td>
<td>56</td>
</tr>
<tr>
<td>5.8 noAdditionalIndent and indentRules</td>
<td>56</td>
</tr>
<tr>
<td>5.8.1 Environments and their arguments</td>
<td>58</td>
</tr>
<tr>
<td>5.8.2 Environments with items</td>
<td>65</td>
</tr>
<tr>
<td>5.8.3 Commands with arguments</td>
<td>66</td>
</tr>
<tr>
<td>5.8.4 ifelsefi code blocks</td>
<td>68</td>
</tr>
<tr>
<td>5.8.5 specialBeginEnd code blocks</td>
<td>70</td>
</tr>
<tr>
<td>5.8.6 afterHeading code blocks</td>
<td>71</td>
</tr>
<tr>
<td>5.8.7 The remaining code blocks</td>
<td>73</td>
</tr>
<tr>
<td>5.8.8 Summary</td>
<td>75</td>
</tr>
<tr>
<td>5.9 Commands and the strings between their arguments</td>
<td>75</td>
</tr>
<tr>
<td>6 The -m (modifylinebreaks) switch</td>
<td>81</td>
</tr>
<tr>
<td>6.1 Text Wrapping</td>
<td>83</td>
</tr>
<tr>
<td>6.1.1 Text wrap: overview</td>
<td>83</td>
</tr>
<tr>
<td>6.1.2 Text wrap: simple examples</td>
<td>84</td>
</tr>
<tr>
<td>6.1.3 Text wrap: blocksFollow examples</td>
<td>85</td>
</tr>
<tr>
<td>6.1.4 Text wrap: blocksBeginWith examples</td>
<td>89</td>
</tr>
<tr>
<td>6.1.5 Text wrap: blocksEndBefore examples</td>
<td>91</td>
</tr>
<tr>
<td>6.1.6 Text wrap: trailing comments and spaces</td>
<td>92</td>
</tr>
<tr>
<td>6.1.7 Text wrap: when before/after</td>
<td>94</td>
</tr>
<tr>
<td>6.1.8 Text wrap: wrapping comments</td>
<td>95</td>
</tr>
<tr>
<td>6.1.9 Text wrap: huge, tabstop and separator</td>
<td>97</td>
</tr>
<tr>
<td>6.2 oneSentencePerLine: modifying line breaks for sentences</td>
<td>98</td>
</tr>
<tr>
<td>6.2.1 oneSentencePerLine: overview</td>
<td>98</td>
</tr>
<tr>
<td>6.2.2 oneSentencePerLine: sentencesFollow</td>
<td>101</td>
</tr>
<tr>
<td>6.2.3 oneSentencePerLine: sentencesBeginWith</td>
<td>102</td>
</tr>
<tr>
<td>6.2.4 oneSentencePerLine: sentencesEndWith</td>
<td>103</td>
</tr>
<tr>
<td>6.2.5 oneSentencePerLine: sentencesDoNOTcontain</td>
<td>104</td>
</tr>
<tr>
<td>6.2.6 Features of the oneSentencePerLine routine</td>
<td>106</td>
</tr>
</tbody>
</table>
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.2.7</td>
<td>oneSentencePerLine: text wrapping and indenting sentences</td>
<td>107</td>
</tr>
<tr>
<td>6.2.8</td>
<td>oneSentencePerLine: text wrapping and indenting sentences, when before/after</td>
<td>110</td>
</tr>
<tr>
<td>6.2.9</td>
<td>oneSentencePerLine: text wrapping sentences and comments</td>
<td>111</td>
</tr>
<tr>
<td>6.3</td>
<td>Poly-switches</td>
<td>111</td>
</tr>
<tr>
<td>6.3.1</td>
<td>Poly-switches for environments</td>
<td>112</td>
</tr>
<tr>
<td>6.3.1.1</td>
<td>Adding line breaks: BeginStartsOnOwnLine and BodyStartsOnOwnLine</td>
<td>112</td>
</tr>
<tr>
<td>6.3.1.2</td>
<td>Adding line breaks: EndStartsOnOwnLine and EndFinishesWithLineBreak</td>
<td>114</td>
</tr>
<tr>
<td>6.3.1.3</td>
<td>poly-switches 1, 2, and 3 only add line breaks when necessary</td>
<td>117</td>
</tr>
<tr>
<td>6.3.1.4</td>
<td>Removing line breaks (poly-switches set to ( -1 ))</td>
<td>117</td>
</tr>
<tr>
<td>6.3.1.5</td>
<td>About trailing horizontal space</td>
<td>119</td>
</tr>
<tr>
<td>6.3.1.6</td>
<td>poly-switch line break removal and blank lines</td>
<td>119</td>
</tr>
<tr>
<td>6.3.2</td>
<td>Poly-switches for double backslash</td>
<td>121</td>
</tr>
<tr>
<td>6.3.2.1</td>
<td>Double backslash starts on own line</td>
<td>121</td>
</tr>
<tr>
<td>6.3.2.2</td>
<td>Double backslash finishes with line break</td>
<td>122</td>
</tr>
<tr>
<td>6.3.2.3</td>
<td>Double backslash poly-switches for specialBeginEnd</td>
<td>123</td>
</tr>
<tr>
<td>6.3.2.4</td>
<td>Double backslash poly-switches for optional and mandatory arguments</td>
<td>123</td>
</tr>
<tr>
<td>6.3.2.5</td>
<td>Double backslash optional square brackets</td>
<td>124</td>
</tr>
<tr>
<td>6.3.3</td>
<td>Poly-switches for other code blocks</td>
<td>125</td>
</tr>
<tr>
<td>6.3.4</td>
<td>Partnering BodyStartsOnOwnLine with argument-based poly-switches</td>
<td>127</td>
</tr>
<tr>
<td>6.3.5</td>
<td>Conflicting poly-switches: sequential code blocks</td>
<td>128</td>
</tr>
<tr>
<td>6.3.6</td>
<td>Conflicting poly-switches: nested code blocks</td>
<td>129</td>
</tr>
<tr>
<td>7</td>
<td>The (-r), (-rv) and (-rr) switches</td>
<td>132</td>
</tr>
<tr>
<td>7.1</td>
<td>Introduction to replacements</td>
<td>132</td>
</tr>
<tr>
<td>7.2</td>
<td>The two types of replacements</td>
<td>133</td>
</tr>
<tr>
<td>7.3</td>
<td>Examples of replacements</td>
<td>133</td>
</tr>
<tr>
<td>8</td>
<td>The (-lines) switch</td>
<td>141</td>
</tr>
<tr>
<td>9</td>
<td>Fine tuning</td>
<td>147</td>
</tr>
<tr>
<td>10</td>
<td>Conclusions and known limitations</td>
<td>156</td>
</tr>
<tr>
<td>11</td>
<td>References</td>
<td>157</td>
</tr>
<tr>
<td>11.1</td>
<td>perl-related links</td>
<td>157</td>
</tr>
<tr>
<td>11.2</td>
<td>conda-related links</td>
<td>157</td>
</tr>
<tr>
<td>11.3</td>
<td>VSCode-related links</td>
<td>157</td>
</tr>
<tr>
<td>11.4</td>
<td>Other links</td>
<td>157</td>
</tr>
<tr>
<td>11.5</td>
<td>Contributors (in chronological order)</td>
<td>158</td>
</tr>
<tr>
<td>A</td>
<td>Required Perl modules</td>
<td>160</td>
</tr>
<tr>
<td>A.1</td>
<td>Module installer script</td>
<td>160</td>
</tr>
<tr>
<td>A.2</td>
<td>Manually installing modules</td>
<td>161</td>
</tr>
<tr>
<td>A.2.1</td>
<td>Linux</td>
<td>161</td>
</tr>
<tr>
<td>A.2.1.1</td>
<td>perlbrew</td>
<td>161</td>
</tr>
<tr>
<td>A.2.1.2</td>
<td>Ubuntu/Debian</td>
<td>161</td>
</tr>
<tr>
<td>A.2.1.3</td>
<td>Ubuntu: using the texlive from apt-get</td>
<td>161</td>
</tr>
<tr>
<td>A.2.1.4</td>
<td>Ubuntu: users without perl</td>
<td>161</td>
</tr>
<tr>
<td>A.2.1.5</td>
<td>Arch-based distributions</td>
<td>161</td>
</tr>
<tr>
<td>A.2.1.6</td>
<td>Alpine</td>
<td>162</td>
</tr>
<tr>
<td>A.2.2</td>
<td>Mac</td>
<td>162</td>
</tr>
<tr>
<td>A.2.3</td>
<td>Windows</td>
<td>162</td>
</tr>
<tr>
<td>A.3</td>
<td>The GCString switch</td>
<td>163</td>
</tr>
<tr>
<td>B</td>
<td>Updating the path variable</td>
<td>164</td>
</tr>
<tr>
<td>B.1</td>
<td>Add to path for Linux</td>
<td>164</td>
</tr>
<tr>
<td>B.2</td>
<td>Add to path for Windows</td>
<td>164</td>
</tr>
</tbody>
</table>
CONTENTS

C Batches of files 166
C.1 location of indent.log ................................................. 166
C.2 interaction with -w switch ......................................... 166
C.3 interaction with -o switch ......................................... 166
C.4 interaction with lines switch ....................................... 166
C.5 interaction with check switches ................................. 167
C.6 when a file does not exist ..................................... 167

D latexindent-yaml-schema.json 168
D.1 VSCode demonstration ............................................. 168

E Using conda 169
E.1 Sample conda installation on Ubuntu ......................... 169

F Using docker 170
F.1 Sample docker installation on Ubuntu ......................... 170
F.2 How to format on Docker ...................................... 170

G pre-commit 171
G.1 Sample pre-commit installation on Ubuntu ................... 171
G.2 pre-commit defaults ............................................. 171
G.3 pre-commit using CPAN ......................................... 172
G.4 pre-commit using conda .......................................... 172
G.5 pre-commit using docker ........................................ 173
G.6 pre-commit example using -l, -m switches ................. 173

H indentconfig options 175
H.1 Why to change the configuration location .................... 175
H.2 How to change the configuration location ................. 176
H.2.1 Linux ....................................................... 176
H.2.2 Windows ................................................... 176
H.2.3 Mac ....................................................... 176

I logFilePreferences 177

J Encoding indentconfig.yaml 178

K dos2unix linebreak adjustment 179

L Differences from Version 2.2 to 3.0 180

List of listings 182

Index 189
SECTION 1

Introduction

1.1 Thanks
I first created latexindent.pl to help me format chapter files in a big project. After I blogged about it on the \TeX{} stack exchange [28] I received some positive feedback and follow-up feature requests. A big thank you to Harish Kumar [2] who helped to develop and test the initial versions of the script.

The YAML-based interface of latexindent.pl was inspired by the wonderful arara tool; any similarities are deliberate, and I hope that it is perceived as the compliment that it is. Thank you to Paulo Cereda and the team for releasing this awesome tool; I initially worried that I was going to have to make a GUI for latexindent.pl, but the release of arara has meant there is no need.

There have been several contributors to the project so far (and hopefully more in the future!); thank you very much to the people detailed in Section 11.5 on page 158 for their valued contributions, and thank you to those who report bugs and request features at [35].

1.2 License
latexindent.pl is free and open source, and it always will be; it is released under the GNU General Public License v3.0.

Before you start using it on any important files, bear in mind that latexindent.pl has the option to overwrite your .tex files. It will always make at least one backup (you can choose how many it makes, see page 28) but you should still be careful when using it. The script has been tested on many files, but there are some known limitations (see Section 10). You, the user, are responsible for ensuring that you maintain backups of your files before running latexindent.pl on them. I think it is important at this stage to restate an important part of the license here:

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

There is certainly no malicious intent in releasing this script, and I do hope that it works as you expect it to; if it does not, please first of all make sure that you have the correct settings, and then feel free to let me know at [35] with a complete minimum working example as I would like to improve the code as much as possible.

**Warning!**
Before you try the script on anything important (like your thesis), test it out on the sample files in the test-case directory [35].

If you have used any version 2.* of latexindent.pl, there are a few changes to the interface; see appendix L on page 180 and the comments throughout this document for details.

1.3 Quick start
If you’d like to get started with latexindent.pl then simply type

```
cmh:~$ latexindent.pl myfile.tex
```

from the command line.
We give an introduction to \texttt{latexindent.pl} using Listing 1; there is no explanation in this section, which is deliberate for a quick start. The rest of the manual is more verbose.

\begin{tabular}{|l|}
\hline
\textbf{LISTING 1: quick-start.tex} \\
\begin{verbatim}
\documentclass{article} \\
\usepackage[
inner=2.5cm, \\
]{geometry} \\
\begin{document} \\
A quick start \\
demonstration for latexindent.pl. \\
\begin{myenv} \\
The body of environments and \\
other code blocks \\
receive indentation. \\
\end{myenv} \\
\end{document}
\end{verbatim} \\
\hline
\end{tabular}

Running
\begin{verbatim}
$ latexindent.pl quick-start.tex
\end{verbatim}
gives Listing 2.

\begin{tabular}{|l|}
\hline
\textbf{LISTING 2: quick-start-default.tex} \\
\begin{verbatim}
\documentclass{article} \\
\usepackage[
inner=2.5cm, \\
]{geometry} \\
\begin{document} \\
A quick start \\
demonstration for latexindent.pl. \\
\begin{myenv} \\
The body of environments and \\
other code blocks \\
receive indentation. \\
\end{myenv} \\
\end{document}
\end{verbatim} \\
\hline
\end{tabular}

\textbf{example 1}

Running
\begin{verbatim}
$ latexindent.pl -l quick-start1.yaml quick-start.tex
\end{verbatim}
gives Listing 3.
1.3 Quick start

Listing 3: quick-start-mod1.tex
```
\documentclass{article}
\usepackage[
inner=2.5cm,
]{geometry}
\begin{document}
A quick start
demonstration for latexindent.pl.
\begin{myenv}
The body of environments and
other code blocks
receive indentation.
\end{myenv}
\end{document}
```

See Section 5.4.

Example 2

Running
```
cmh:~$ latexindent.pl -l quick-start2.yaml quick-start.tex
```
gives Listing 5.

Listing 5: quick-start-mod2.tex
```
\documentclass{article}
\usepackage[
inner=2.5cm,
]{geometry}
\begin{document}
A quick start
demonstration for latexindent.pl.
\begin{myenv}
\quad The body of environments and
\quad other code blocks
\quad receive indentation.
\end{myenv}
\end{document}
```

See Section 5.8.

Example 3

Running
```
cmh:~$ latexindent.pl -l quick-start3.yaml quick-start.tex
```
gives Listing 7.
1.3 Quick start

**Listing 7: quick-start-mod3.tex**

```latex
\documentclass{article}
\usepackage[inner=2.5cm,]{geometry}
\begin{document}
A quick start
demonstration for latexindent.pl.
\begin{myenv}
The body of environments and
other code blocks
receive indentation.
\end{myenv}
\end{document}
```

See Section 5.8.

**Example 4**

Running

```
cmh:~$ latexindent.pl -m -l quick-start4.yaml quick-start.tex
```

gives Listing 9.

**Listing 9: quick-start-mod4.tex**

```latex
\documentclass{article}
\usepackage[inner=2.5cm,]{geometry}
\begin{document}
A quick start
demonstration for latexindent.pl.
\begin{myenv}
The body of environments and
other code blocks
receive indentation.
\end{myenv}
\end{document}
```

Full details of text wrapping in Section 6.1.

**Example 5**

Running

```
cmh:~$ latexindent.pl -m -l quick-start5.yaml quick-start.tex
```

gives Listing 11.

**Listing 10: quick-start4.yaml**

```yaml
noAdditionalIndent:
  myenv: 1
```

**Listing 10: quick-start4.yaml**

```yaml
modifyLineBreaks:
  textWrapOptions:
    columns: 20
```
1.3 Quick start

example 6

Running

cmh:~$ latexindent.pl -m -l quick-start6.yaml quick-start.tex

gives Listing 13.

example 7

Running

cmh:~$ latexindent.pl -m -l quick-start7.yaml quick-start.tex

gives Listing 15.
1.3 Quick start

\begin{listing}
\begin{verbatim}
documentclass{article}
\usepackage[
    inner=2.5cm,
]{geometry}
\begin{document}
A quick start
demonstration for latexindent.pl.
\begin{myenv}
The body of environments and
other code blocks
receive indentation.\end{myenv}
\end{document}
\end{verbatim}
\end{listing}

Full details of \textit{poly-switches} are covered in Section 6.3.

\begin{example}
Running
\begin{verbatim}
cmh:\textit{$latexindent$.pl -l quick-start8.yaml quick-start.tex}
\end{verbatim}
gives Listing 17; note that the \textit{preamble} has been indented.

\begin{listing}
\begin{verbatim}
documentclass{article}
\usepackage[
    inner=2.5cm,
]{geometry}
\begin{document}
A quick start
demonstration for latexindent.pl.
\begin{myenv}
The body of environments and
other code blocks
receive indentation.\end{myenv}
\end{document}
\end{verbatim}
\end{listing}

See Section 5.3.
\end{example}

\begin{example}
Running
\begin{verbatim}
cmh:\textit{$latexindent$.pl -l quick-start9.yaml quick-start.tex}
\end{verbatim}
gives Listing 19.
1.4 Required perl modules

If you receive an error message such as that given in Listing 21, then you need to install the missing perl modules.

Listing 21: Possible error messages

```
Can't locate File/HomeDir.pm in @INC (@INC contains: /Library/Perl/5.12/darwin-thread-multi-2level/
/Library/Perl/5.12/
/Network/Library/Perl/5.12/darwin-thread-multi-2level/
/Network/Library/Perl/5.12/
/Library/Perl/Updates/5.12.4/darwin-thread-multi-2level/
/Library/Perl/Updates/5.12.4/
/System/Library/Perl/5.12/darwin-thread-multi-2level/
/System/Library/Perl/5.12/
/System/Library/Perl/Extras/5.12.4.darwin-thread-multi-2level/
/System/Library/Perl/Extras/5.12.4/)
BEGIN failed--compilation aborted at helloworld.pl line 10.
```

latexindent.pl ships with a script to help with this process; if you run the following script, you should be prompted to install the appropriate modules.

```
cmh:~$ perl latexindent-module-installer.pl
```

You might also like to see https://stackoverflow.com/questions/19590042/error-cant-locate-file-home-dir-pm-in-inc, for example, as well as appendix A on page 160.

1.5 About this documentation

As you read through this documentation, you will see many listings; in this version of the documentation, there are a total of 621. This may seem a lot, but I deem it necessary in presenting the various different options of latexindent.pl and the associated output that they are capable of producing.

The different listings are presented using different styles:

Listing 22: demo-tex.tex

demonstration .tex file

Listing 23:

```
fileExtensionPreference
```

47 fileExtensionPreference:
48 .tex: 1
49 .sty: 2
50 .cls: 3
51 .bib: 4

This type of listing is a .tex file.

This type of listing is a .yaml file; when you see line numbers given (as here) it means that the snippet is taken directly from defaultSettings.yaml, discussed in detail in Section 5 on page 27.
1.6  A word about regular expressions

As you read this documentation, you may encounter the term *regular expressions*. I've tried to write this documentation in such a way so as to allow you to engage with them or not, as you prefer. This documentation is not designed to be a guide to regular expressions, and if you'd like to read about them, I recommend [34].
SECTION 2

Demonstration: before and after

Let’s give a demonstration of some before and after code – after all, you probably won’t want to try the script if you don’t much like the results. You might also like to watch the video demonstration I made on youtube [48]

As you look at Listings 26 to 31, remember that latexindent.pl is just following its rules, and there is nothing particular about these code snippets. All of the rules can be modified so that you can personalise your indentation scheme.

In each of the samples given in Listings 26 to 31 the ‘before’ case is a ‘worst case scenario’ with no effort to make indentation. The ‘after’ result would be the same, regardless of the leading white space at the beginning of each line which is stripped by latexindent.pl (unless a verbatim-like environment or noIndentBlock is specified – more on this in Section 5).

**LISTING 26: filecontents1.tex**

```latex
\begin{filecontents}{mybib.bib}
@online{strawberryperl,  
title="Strawberry Perl",  
url="http://strawberryperl.com/"}
@online{cmhblog,  
title="A Perl script ...
url="...  
}
\end{filecontents}
```

**LISTING 27: filecontents1.tex default output**

```latex
\begin{filecontents}{mybib.bib}
@online{strawberryperl,  
title="Strawberry Perl",  
url="http://strawberryperl.com/"}
@online{cmhblog,  
title="A Perl script ...
url="...  
}
\end{filecontents}
```

**LISTING 28: tikzset.tex**

```latex
\tikzset{  
shrink inner sep/.code={  
\pgfkeysgetvalue...  
\pgfkeysgetvalue...
}  }
```

**LISTING 29: tikzset.tex default output**

```latex
\tikzset{  
shrink inner sep/.code={  
\pgfkeysgetvalue...  
\pgfkeysgetvalue...
}  }
```

**LISTING 30: pstricks.tex**

```latex
\def\Picture#1{%  
\def\stripH{#1}%  \begin{pspicture}[showgrid]  \psforeach\row{%  {{3,2.8,2.7,3,3.1},%,  {2.8,1,1.2,2,3},%  ...  \}%  \expandafter...  }  \end{pspicture}
```

**LISTING 31: pstricks.tex default output**

```latex
\def\Picture#1{%  \def\stripH{#1}%  \begin{pspicture}[showgrid]  \psforeach\row{%  {{3,2.8,2.7,3,3.1},%,  {2.8,1,1.2,2,3},%  ...  \}%  \expandafter...  }  \end{pspicture}
```
SECTION 3

How to use the script

\texttt{latexindent.pl} ships as part of the \TeX\ Live distribution for Linux and Mac users; \texttt{latexindent.exe} ships as part of the \TeX\ Live for Windows users. These files are also available from github \cite{github} should you wish to use them without a \TeX\ distribution; in this case, you may like to read appendix \ref{appendix:b} on page \pageref{appendix:b} which details how the \texttt{path} variable can be updated.

In what follows, we will always refer to \texttt{latexindent.pl}, but depending on your operating system and preference, you might substitute \texttt{latexindent.exe} or simply \texttt{latexindent}.

There are two ways to use \texttt{latexindent.pl}: from the command line, and using arara; we discuss these in Section \ref{section:3.2} and Section \ref{section:3.3} respectively. We will discuss how to change the settings and behaviour of the script in Section \ref{section:5} on page \pageref{section:5}.

3.1 Requirements

3.1.1 Perl users

Perl users will need a few standard Perl modules – see appendix \ref{appendix:a} on page \pageref{appendix:a} for details; in particular, note that a module installer helper script is shipped with \texttt{latexindent.pl}.

3.1.2 Windows users without perl

\texttt{latexindent.pl} ships with \texttt{latexindent.exe} for Windows users, so that you can use the script with or without a Perl distribution.

\texttt{latexindent.exe} is available from \cite{github}.

MiKTeX users on Windows may like to see \cite{miktex} for details of how to use \texttt{latexindent.exe} without a Perl installation.

3.1.3 Ubuntu Linux users without perl

\texttt{latexindent.pl} ships with \texttt{latexindent-linux} for Ubuntu Linux users, so that you can use the script with or without a Perl distribution.

\texttt{latexindent-linux} is available from \cite{github}.

3.1.4 macOS users without perl

\texttt{latexindent.pl} ships with \texttt{latexindent-macos} for macOS users, so that you can use the script with or without a Perl distribution.

\texttt{latexindent-macos} is available from \cite{github}.

3.1.5 conda users

Users of conda should see the details given in appendix \ref{appendix:e}.

3.1.6 docker users

Users of docker should see the details given in appendix \ref{appendix:f}.
3.2 From the command line

latexindent.pl has a number of different switches/flags/options, which can be combined in any way that you like, either in short or long form as detailed below. latexindent.pl produces a .log file, indent.log, every time it is run; the name of the log file can be customised, but we will refer to the log file as indent.log throughout this document. There is a base of information that is written to indent.log, but other additional information will be written depending on which of the following options are used.

-v, --version

```latexindent.pl -v
latexindent.pl --version```

This will output only the version number to the terminal.

-vv, --vversion

```latexindent.pl -vv
latexindent.pl --vversion```

This will output verbose version details to the terminal, including the location of latexindent.pl and defaultSettings.yaml.

-h, --help

```latexindent.pl -h
latexindent.pl --help```

As above this will output a welcome message to the terminal, including the version number and available options.

```latexindent.pl myfile.tex```

This will operate on myfile.tex, but will simply output to your terminal; myfile.tex will not be changed by latexindent.pl in any way using this command.

You can instruct latexindent.pl to operate on multiple (batches) of files, for example

```latexindent.pl myfile1.tex myfile2.tex```

Full details are given in appendix C on page 166.

-w, --overwrite

```latexindent.pl -w myfile.tex
latexindent.pl --overwrite myfile.tex
latexindent.pl myfile.tex --overwrite```

This will overwrite myfile.tex, but it will make a copy of myfile.tex first. You can control the name of the extension (default is .bak), and how many different backups are made – more on this in Section 5, and in particular see backupExtension and onlyOneBackUp.

Note that if latexindent.pl can not create the backup, then it will exit without touching your original file; an error message will be given asking you to check the permissions of the backup file.

-wd, --overwriteIfDifferent
3.2 From the command line

```
• cmh:~$ latexindent.pl -wd myfile.tex
• cmh:~$ latexindent.pl --overwriteIfDifferent myfile.tex
• cmh:~$ latexindent.pl myfile.tex --overwriteIfDifferent
```

This will overwrite myfile.tex but only if the indented text is different from the original. If the indented text is not different from the original, then myfile.tex will not be overwritten.

All other details from the -w switch are relevant here. If you call latexindent.pl with both the -wd and the -w switch, then the -w switch will be deactivated and the -wd switch takes priority.

-o=output.tex, -outputfile=output.tex

```
• cmh:~$ latexindent.pl -o=output.tex myfile.tex
• cmh:~$ latexindent.pl myfile.tex -o=output.tex
• cmh:~$ latexindent.pl --outputfile=output.tex myfile.tex
• cmh:~$ latexindent.pl --outputfile output.tex myfile.tex
```

This will indent myfile.tex and output it to output.tex, overwriting it (output.tex) if it already exists¹.

Note that if latexindent.pl is called with both the -w and -o switches, then -w will be ignored and -o will take priority (this seems safer than the other way round). The same is true for the -wd switch, and the -o switch takes priority over it.

Note that using -o as above is equivalent to using

```
• cmh:~$ latexindent.pl myfile.tex > output.tex
```

You can call the -o switch with the name of the output file without an extension; in this case, latexindent.pl will use the extension from the original file. For example, the following two calls to latexindent.pl are equivalent:

```
• cmh:~$ latexindent.pl myfile.tex -o=output
• cmh:~$ latexindent.pl myfile.tex -o=output.tex
```

You can call the -o switch using a + symbol at the beginning; this will concatenate the name of the input file and the text given to the -o switch. For example, the following two calls to latexindent.pl are equivalent:

```
• cmh:~$ latexindent.pl myfile.tex -o=new
• cmh:~$ latexindent.pl myfile.tex -o=myfilenew.tex
```

You can call the -o switch using a ++ symbol at the end of the name of your output file; this tells latexindent.pl to search successively for the name of your output file concatenated with 0, 1, … while the name of the output file exists. For example,

```
• cmh:~$ latexindent.pl myfile.tex -o=output++
```

tells latexindent.pl to output to output0.tex, but if it exists then output to output1.tex, and so on.

Calling latexindent.pl with simply

```
• cmh:~$ latexindent.pl myfile.tex -o=++
```

¹Users of version 2.+ should note the subtle change in syntax
3.2 From the command line

```
$ latexindent.pl myfile.tex -o=+out++
```

tells latexindent.pl to output to myfileout0.tex, but if it exists, then try myfileout1.tex, and so on.

The + and ++ feature of the -o switch can be combined; for example, calling

```
$ latexindent.pl myfile.tex -o=+out++.tex
```

See appendix I on page 180 for details of how the interface has changed from Version 2.2 to Version 3.0 for this flag.

-s, -silent

```
$ latexindent.pl -s myfile.tex
$ latexindent.pl myfile.tex -s
```

Silent mode: no output will be given to the terminal.

-t, -trace

```
$ latexindent.pl -t myfile.tex
$ latexindent.pl myfile.tex -t
```

Tracing mode: verbose output will be given to indent.log. This is useful if latexindent.pl has made a mistake and you're trying to find out where and why. You might also be interested in learning about latexindent.pl’s thought process – if so, this switch is for you, although it should be noted that, especially for large files, this does affect performance of the script.

-tt, -tttrace

```
$ latexindent.pl -tt myfile.tex
$ latexindent.pl myfile.tex -tt
```

More detailed tracing mode: this option gives more details to indent.log than the standard trace option (note that, even more so than with -t, especially for large files, performance of the script will be affected).

-l, -local[=myyaml.yaml,other.yaml,...]

```
$ latexindent.pl -l myfile.tex
$ latexindent.pl -l=myyaml.yaml myfile.tex
$ latexindent.pl -l first.yaml,second.yaml,third.yaml myfile.tex
$ latexindent.pl -l=first.yaml,second.yaml,third.yaml myfile.tex
$ latexindent.pl myfile.tex -l=first.yaml,second.yaml,third.yaml
```

latexindent.pl will always load defaultSettings.yaml (rhymes with camel) and if it is called with the -l switch and it finds localSettings.yaml in the same directory as myfile.tex, then, if not found, it looks for localSettings.yaml (and friends, see Section 4.2 on page 24) in the current working directory, then these settings will be added to the indentation scheme. Information will be given in indent.log on the success or failure of loading localSettings.yaml.
The `-l` flag can take an optional parameter which details the name (or names separated by commas) of a YAML file(s) that resides in the same directory as `myfile.tex`; you can use this option if you would like to load a settings file in the current working directory that is *not* called `localSettings.yaml`. In fact, you can specify both relative and absolute paths for your YAML files; for example:

```
cmh:~$ latexindent.pl -l=../../myyaml.yaml myfile.tex
cmh:~$ latexindent.pl -l=/home/cmhughes/Desktop/myyaml.yaml myfile.tex
cmh:~$ latexindent.pl -l=C:\Users\cmhughes\Desktop\myyaml.yaml myfile.tex
```

You will find a lot of other explicit demonstrations of how to use the `-l` switch throughout this documentation.

You can call the `-l` switch with a `+' symbol either before or after another YAML file; for example:

```
cmh:~$ latexindent.pl -l=+myyaml.yaml myfile.tex
```

which translate, respectively, to

```
cmh:~$ latexindent.pl -l=localSettings.yaml,myyaml.yaml myfile.tex
```

Note that the following is *not* allowed:

```
cmh:~$ latexindent.pl -l+=myyaml.yaml myfile.tex
```

and

```
cmh:~$ latexindent.pl -l + myyaml.yaml myfile.tex
```

will *only* load `localSettings.yaml`, and `myyaml.yaml` will be ignored. If you wish to use spaces between any of the YAML settings, then you must wrap the entire list of YAML files in quotes, as demonstrated above.

You may also choose to omit the `.yaml` extension, such as

```
cmh:~$ latexindent.pl -l=localSettings,myyaml myfile.tex
```

### `-y, --yaml=yaml settings`

```
cmh:~$ latexindent.pl myfile.tex -y="defaultIndent:'\t'"
cmh:~$ latexindent.pl myfile.tex -y="defaultIndent:'\t',maximumIndentation:'\t'"
cmh:~$ latexindent.pl myfile.tex -y="indentRules:one:'\t\t\t\t'"
cmh:~$ latexindent.pl myfile.tex
   -y='modifyLineBreaks:environments:EndStartsOnOwnLine:3' -m
cmh:~$ latexindent.pl myfile.tex
   -y='modifyLineBreaks:environments:one:EndStartsOnOwnLine:3' -m
```

You can specify YAML settings from the command line using the `-y` or `--yaml` switch; sample demonstrations are given above. Note, in particular, that multiple settings can be specified by separating them via commas. There is a further option to use a `;` to separate fields, which is demonstrated in Section 4.3 on page 25.
Any settings specified via this switch will be loaded after any specified using the -l switch. This is discussed further in Section 4.4 on page 25.

-d, --onlydefault

```
cmh:~$ latexindent.pl -d myfile.tex
```

Only defaultSettings.yaml: you might like to read Section 5 before using this switch. By default, latexindent.pl will always search for indentconfig.yaml or .indentconfig.yaml in your home directory. If you would prefer it not to do so then (instead of deleting or renaming indentconfig.yaml or .indentconfig.yaml) you can simply call the script with the -d switch; note that this will also tell the script to ignore localSettings.yaml even if it has been called with the -l switch; latexindent.pl will also ignore any settings specified from the -y switch.

-c, --cruft=<directory>

```
cmh:~$ latexindent.pl -c=/path/to/directory/ myfile.tex
```

If you wish to have backup files and indent.log written to a directory other than the current working directory, then you can send these 'cruft' files to another directory. Note the use of a trailing forward slash.

If the cruft directory does not exist, latexindent.pl will attempt to create it.

-g, --logfile=<name of log file>

```
cmh:~$ latexindent.pl -g=other.log myfile.tex
```

By default, latexindent.pl reports information to indent.log, but if you wish to change the name of this file, simply call the script with your chosen name after the -g switch as demonstrated above.

If latexindent.pl can not open the log file that you specify, then the script will operate, and no log file will be produced; this might be helpful to users who wish to specify the following, for example

```
cmh:~$ latexindent.pl -g /dev/null myfile.tex
```

-sl, --screenlog

```
cmh:~$ latexindent.pl -sl myfile.tex
```

Using this option tells latexindent.pl to output the log file to the screen, as well as to your chosen log file.

-m, --modifylinebreaks

```
cmh:~$ latexindent.pl -m myfile.tex
```

One of the most exciting developments in Version 3.0 is the ability to modify line breaks; for full details see Section 6 on page 81

latexindent.pl can also be called on a file without the file extension, for example
3.2 From the command line

and in which case, you can specify the order in which extensions are searched for; see Listing 36 on page 27 for full details.

STDIN

```
cmh:~$ latexindent.pl myfile
```

latexindent.pl will allow input from STDIN, which means that you can pipe output from other commands directly into the script. For example assuming that you have content in myfile.tex, then the above command will output the results of operating upon myfile.tex.

If you wish to use this feature with your own local settings, via the -l switch, then you should finish your call to latexindent.pl with a - sign:

```
cmh:~$ cat myfile.tex | latexindent.pl
```

Similarly, if you simply type latexindent.pl at the command line, then it will expect (STDIN) input from the command line.

```
cmh:~$ latexindent.pl
```

Once you have finished typing your input, you can press

- • CTRL+D on Linux
- • CTRL+Z followed by ENTER on Windows

to signify that your input has finished. Thanks to [9] for an update to this feature.

- r, -replacement

```
cmh:~$ latexindent.pl -r myfile.tex
```

You can call latexindent.pl with the -r switch to instruct it to perform replacements/substitutions on your file; full details and examples are given in Section 7 on page 132.

- rv, -replacementrespectverb

```
cmh:~$ latexindent.pl -rv myfile.tex
```

You can instruct latexindent.pl to perform replacements/substitutions by using the -rv switch, but will respect verbatim code blocks; full details and examples are given in Section 7 on page 132.

- rr, -onlyreplacement

```
cmh:~$ latexindent.pl -rr myfile.tex
```

You can instruct latexindent.pl to skip all of its other indentation operations and only perform replacements/substitutions by using the -rr switch; full details and examples are given in Section 7 on page 132.

- k, -check
You can instruct `latexindent.pl` to check if the text after indentation matches that given in the original file.

The exit code of `latexindent.pl` is 0 by default. If you use the `-k` switch then

- if the text after indentation matches that given in the original file, then the exit code is 0;
- if the text after indentation does *not* match that given in the original file, then the exit code is 1.

The value of the exit code may be important to those wishing to, for example, check the status of the indentation in continuous integration tools such as GitHub Actions. Full details of the exit codes of `latexindent.pl` are given in Table 1.

A simple diff will be given in `indent.log`.

**-kv, --checkv**

You can instruct `latexindent.pl` to check if the text after indentation matches that given in the original file.

The exit code of `latexindent.pl` is 0 by default. If you use the `-k` switch then

- if the text after indentation matches that given in the original file, then the exit code is 0;
- if the text after indentation does *not* match that given in the original file, then the exit code is 1.

The value of the exit code may be important to those wishing to, for example, check the status of the indentation in continuous integration tools such as GitHub Actions. Full details of the exit codes of `latexindent.pl` are given in Table 1.

A simple diff will be given in `indent.log`.

**-n, --lines=MIN-MAX**

The lines switch instructs `latexindent.pl` to operate only on specific line ranges within `myfile.tex`.

Complete demonstrations are given in Section 8.

**--GCString**

instructs `latexindent.pl` to load the `Unicode::GCString` module. This should only be necessary if you find that the alignment at ampersand routine (described in Section 5.5) does not work for your language. Further details are given in appendix A.3.

### 3.3 From arara

Using `latexindent.pl` from the command line is fine for some folks, but others may find it easier to use from `arara`; you can find the `arara` rule for `latexindent.pl` and its associated documentation at [1].

### 3.4 Summary of exit codes

Assuming that you call `latexindent.pl` on `myfile.tex`

then `latexindent.pl` can exit with the exit codes given in Table 1.
## 3.4 Summary of exit codes

### Table 1: Exit codes for latexindent.pl

<table>
<thead>
<tr>
<th>exit code</th>
<th>indentation</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>✔</td>
<td>success; if \texttt{-k} or \texttt{-kv} active, indented text matches original</td>
</tr>
<tr>
<td>0</td>
<td>✗</td>
<td>success; if \texttt{-version}, \texttt{-version} or \texttt{-help}, no indentation performed</td>
</tr>
<tr>
<td>1</td>
<td>✔</td>
<td>success; and \texttt{-k} or \texttt{-kv} active; indented text different from original</td>
</tr>
<tr>
<td>2</td>
<td>✗</td>
<td>failure, \texttt{defaultSettings.yaml} could not be read</td>
</tr>
<tr>
<td>3</td>
<td>✗</td>
<td>failure, \texttt{myfile.tex} not found</td>
</tr>
<tr>
<td>4</td>
<td>✗</td>
<td>failure, \texttt{myfile.tex} exists but cannot be read</td>
</tr>
<tr>
<td>5</td>
<td>✗</td>
<td>failure, \texttt{-w} active, and back-up file cannot be written</td>
</tr>
<tr>
<td>6</td>
<td>✗</td>
<td>failure, \texttt{-c} active, and cruft directory could not be created</td>
</tr>
</tbody>
</table>
SECTION 4

indentconfig.yaml, local settings and the -y switch

The behaviour of latexindent.pl is controlled from the settings specified in any of the YAML files that you tell it to load. By default, latexindent.pl will only load defaultSettings.yaml, but there are a few ways that you can tell it to load your own settings files.

We focus our discussion on indentconfig.yaml, but there are other options which are detailed in appendix H.

4.1 indentconfig.yaml and .indentconfig.yaml

latexindent.pl will always check your home directory for indentconfig.yaml and .indentconfig.yaml (unless it is called with the -d switch), which is a plain text file you can create that contains the absolute paths for any settings files that you wish latexindent.pl to load. There is no difference between indentconfig.yaml and .indentconfig.yaml, other than the fact that .indentconfig.yaml is a 'hidden' file; thank you to [5] for providing this feature. In what follows, we will use indentconfig.yaml, but it is understood that this could equally represent .indentconfig.yaml. If you have both files in existence then indentconfig.yaml takes priority.

For Mac and Linux users, their home directory is /username while Windows (Vista onwards) is C:\Users\username² Listing 32 shows a sample indentconfig.yaml file.

<table>
<thead>
<tr>
<th>Listing 32: indentconfig.yaml (sample)</th>
</tr>
</thead>
<tbody>
<tr>
<td># Paths to user settings for latexindent.pl</td>
</tr>
<tr>
<td>#</td>
</tr>
<tr>
<td># Note that the settings will be read in the order you</td>
</tr>
<tr>
<td># specify here- each successive settings file will overwrite</td>
</tr>
<tr>
<td># the variables that you specify</td>
</tr>
<tr>
<td>paths:</td>
</tr>
<tr>
<td>- /home/cmhughes/Documents/yamlfiles/mysettings.yaml</td>
</tr>
<tr>
<td>- /home/cmhughes/folder/othersettings.yaml</td>
</tr>
<tr>
<td>- /some/other/folder/anynameyouwant.yaml</td>
</tr>
<tr>
<td>- C:\Users\chughes\Documents\mysettings.yaml</td>
</tr>
<tr>
<td>- C:\Users\chughes\Desktop\test spaces\more spaces.yaml</td>
</tr>
</tbody>
</table>

Note that the .yaml files you specify in indentconfig.yaml will be loaded in the order in which you write them. Each file doesn’t have to have every switch from defaultSettings.yaml; in fact, I recommend that you only keep the switches that you want to change in these settings files.

To get started with your own settings file, you might like to save a copy of defaultSettings.yaml in another directory and call it, for example, mysettings.yaml. Once you have added the path to indentconfig.yaml you can change the switches and add more code-block names to it as you see fit – have a look at Listing 33 for an example that uses four tabs for the default indent, adds the tabbing environment/command to the list of environments that contains alignment delimiters; you might also like to refer to the many YAML files detailed throughout the rest of this documentation.

²If you’re not sure where to put indentconfig.yaml, don’t worry latexindent.pl will tell you in the log file exactly where to put it assuming it doesn’t exist already.
4.2 localSettings.yaml and friends

The `-l` switch tells latexindent.pl to look for localSettings.yaml and/or friends in the same directory as myfile.tex. For example, if you use the following command

```
cmh:~$ latexindent.pl -l myfile.tex
```

then latexindent.pl will search for and then, assuming they exist, load each of the following files in the following order:

1. localSettings.yaml
2. latexindent.yaml
3. .localSettings.yaml
4. .latexindent.yaml

These files will be assumed to be in the same directory as myfile.tex, or otherwise in the current working directory. You do not need to have all of the above files, usually just one will be sufficient. In what follows, whenever we refer to localSettings.yaml it is assumed that it can mean any of the four named options listed above.

If you'd prefer to name your localSettings.yaml file something different, (say, mysettings.yaml as in Listing 33) then you can call latexindent.pl using, for example,

```
cmh:~$ latexindent.pl -l mysettings.yaml
```

Warning! When editing .yaml files it is extremely important to remember how sensitive they are to spaces. I highly recommend copying and pasting from defaultSettings.yaml when you create your first whatevernameyoulike.yaml file. If latexindent.pl can not read your .yaml file it will tell you so in indent.log.

You can make sure that your settings are loaded by checking indent.log for details – if you have specified a path that latexindent.pl doesn't recognise then you'll get a warning, otherwise you'll get confirmation that latexindent.pl has read your settings file.

If you find that latexindent.pl does not read your YAML file, then it might be as a result of the default commandline encoding not being UTF-8; normally this will only occur for Windows users. In this case, you might like to explore the encoding option for indentconfig.yaml as demonstrated in Listing 34.

```
LISTING 34: The encoding option for indentconfig.yaml

encoding: GB2312
paths:
- D:\cmh\latexindent.yaml
```

Thank you to [15] for this contribution; please see appendix J on page 178 and details within [42] for further information.

---

4.2 localSettings.yaml and friends

The `-l` switch tells latexindent.pl to look for localSettings.yaml and/or friends in the same directory as myfile.tex. For example, if you use the following command:

```
cmh:~$ latexindent.pl -l myfile.tex
```

then latexindent.pl will search for and then, assuming they exist, load each of the following files in the following order:

1. localSettings.yaml
2. latexindent.yaml
3. .localSettings.yaml
4. .latexindent.yaml

These files will be assumed to be in the same directory as myfile.tex, or otherwise in the current working directory. You do not need to have all of the above files, usually just one will be sufficient. In what follows, whenever we refer to localSettings.yaml it is assumed that it can mean any of the four named options listed above.

If you'd prefer to name your localSettings.yaml file something different, (say, mysettings.yaml as in Listing 33) then you can call latexindent.pl using, for example,

```
cmh:~$ latexindent.pl -l mysettings.yaml
```

---

3Windows users may find that they have to end .yaml files with a blank line
Any settings file(s) specified using the -l switch will be read after defaultSettings.yaml and, assuming they exist, any user setting files specified in indentconfig.yaml.

Your settings file can contain any switches that you’d like to change; a sample is shown in Listing 35, and you’ll find plenty of further examples throughout this manual.

**LISTING 35: localSettings.yaml (example)**

```yaml
# verbatim environments - environments specified
# here will not be changed at all!
verbatimEnvironments:
  cmhenvironment: 0
  myenv: 1
```

You can make sure that your settings file has been loaded by checking indent.log for details; if it can not be read then you receive a warning, otherwise you'll get confirmation that latexindent.pl has read your settings file.

### 4.3 The -y|yaml switch

You may use the -y switch to load your settings; for example, if you wished to specify the settings from Listing 35 using the -y switch, then you could use the following command:

```bash
cmh:~$ latexindent.pl -y="verbatimEnvironments:cmhenvironment:0;myenv:1" myfile.tex
```

Note the use of ; to specify another field within verbatimEnvironments. This is shorthand, and equivalent, to using the following command:

```bash
cmh:~$ latexindent.pl
   -y="verbatimEnvironments:cmhenvironment:0,verbatimEnvironments:myenv:1"
   myfile.tex
```

You may, of course, specify settings using the -y switch as well as, for example, settings loaded using the -l switch; for example,

```bash
cmh:~$ latexindent.pl
   -l=mysettings.yaml
   -y="verbatimEnvironments:cmhenvironment:0;myenv:1" myfile.tex
```

Any settings specified using the -y switch will be loaded after any specified using indentconfig.yaml and the -l switch.

If you wish to specify any regex-based settings using the -y switch, it is important not to use quotes surrounding the regex; for example, with reference to the 'one sentence per line' feature (Section 6.2 on page 98) and the listings within Listing 376 on page 101, the following settings give the option to have sentences end with a semicolon

```bash
cmh:~$ latexindent.pl
   --yaml='modifyLineBreaks:oneSentencePerLine:sentencesEndWith:other:\;'
```

### 4.4 Settings load order

latexindent.pl loads the settings files in the following order:

1. defaultSettings.yaml is always loaded, and can not be renamed;
2. anyUserSettings.yaml and any other arbitrarily-named files specified in indentconfig.yaml;
3. `localSettings.yaml` but only if found in the same directory as `myfile.tex` and called with `-l` switch; this file can be renamed, provided that the call to `latexindent.pl` is adjusted accordingly (see Section 4.2). You may specify both relative and absolute paths to other YAML files using the `-l` switch, separating multiple files using commas;

4. any settings specified in the `-y` switch.

A visual representation of this is given in Figure 1.

![Figure 1: Schematic of the load order described in Section 4.4; solid lines represent mandatory files, dotted lines represent optional files. `indentconfig.yaml` can contain as many files as you like. The files will be loaded in order; if you specify settings for the same field in more than one file, the most recent takes priority.](image-url)
SECTION 5

defaultSettings.yaml

latexindent.pl loads its settings from defaultSettings.yaml. The idea is to separate the behavior of the script from the internal working – this is very similar to the way that we separate content from form when writing our documents in \LaTeX.

If you look in defaultSettings.yaml you’ll find the switches that govern the behaviour of latexindent.pl. If you’re not sure where defaultSettings.yaml resides on your computer, don’t worry as indent.log will tell you where to find it. defaultSettings.yaml is commented, but here is a description of what each switch is designed to do. The default value is given in each case; whenever you see integer in this section, assume that it must be greater than or equal to 0 unless otherwise stated.

For most of the settings in defaultSettings.yaml that are specified as integers, then we understand 0 to represent ‘off’ and 1 to represent ‘on’. For fields that allow values other than 0 or 1, it is hoped that the specific context and associated commentary should make it clear which values are allowed.

**fileExtensionPreference: (fields)**

latexindent.pl can be called to act on a file without specifying the file extension. For example we can call

```cmsh~$ latexindent.pl myfile```

in which case the script will look for myfile with the extensions specified in fileExtensionPreference in their numeric order. If no match is found, the script will exit. As with all of the fields, you should change and/or add to this as necessary.

**Listing 36: fileExtensionPreference**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>47</td>
<td>fileExtensionPreference:</td>
</tr>
<tr>
<td>48</td>
<td>.tex: 1</td>
</tr>
<tr>
<td>49</td>
<td>.sty: 2</td>
</tr>
<tr>
<td>50</td>
<td>.cls: 3</td>
</tr>
<tr>
<td>51</td>
<td>.bib: 4</td>
</tr>
</tbody>
</table>

Calling latexindent.pl myfile with the (default) settings specified in Listing 36 means that the script will first look for myfile.tex, then myfile.sty, myfile.cls, and finally myfile.bib in order.

5.1 Backup and log file preferences

**backupExtension: (extension name)**

If you call latexindent.pl with the -w switch (to overwrite myfile.tex) then it will create a backup file before doing any indentation; the default extension is .bak, so, for example, myfile.bak0 would be created when calling latexindent.pl myfile.tex for the first time.

By default, every time you subsequently call latexindent.pl with the -w to act upon myfile.tex, it will create successive back up files: myfile.bak1, myfile.bak2, etc.

### Throughout this manual, listings shown with line numbers represent code taken directly from defaultSettings.yaml.
5.1 Backup and log file preferences

**onlyOneBackUp**: (integer)

If you don’t want a backup for every time that you call `latexindent.pl` (so you don’t want `myfile.bak1`, `myfile.bak2`, etc) and you simply want `myfile.bak` (or whatever you chose `backupExtension` to be) then change `onlyOneBackUp` to 1; the default value of `onlyOneBackUp` is 0.

**maxNumberOfBackUps**: (integer)

Some users may only want a finite number of backup files, say at most 3, in which case, they can change this switch. The smallest value of `maxNumberOfBackUps` is 0 which will not prevent backup files being made; in this case, the behaviour will be dictated entirely by `onlyOneBackUp`. The default value of `maxNumberOfBackUps` is 0.

**cycleThroughBackUps**: (integer)

Some users may wish to cycle through backup files, by deleting the oldest backup file and keeping only the most recent; for example, with `maxNumberOfBackUps`: 4, and `cycleThroughBackUps` set to 1 then the `copy` procedure given below would be obeyed.

```
cmh:\$ copy myfile.bak1 to myfile.bak0
cmh:\$ copy myfile.bak2 to myfile.bak1
```

The default value of `cycleThroughBackUps` is 0.

**logFilePreferences**: (fields)

`latexindent.pl` writes information to `indent.log`, some of which can be customized by changing `logFilePreferences`; see Listing 37. If you load your own user settings (see Section 4 on page 23) then `latexindent.pl` will detail them in `indent.log`; you can choose not to have the details logged by switching `showEveryYamlRead` to 0. Once all of your settings have been loaded, you can see the amalgamated settings in the log file by switching `showAmalgamatedSettings` to 1, if you wish.

```
Listing 37: logFilePreferences

logFilePreferences:
  showEveryYamlRead: 1
  showAmalgamatedSettings: 0
  showDecorationStartCodeBlockTrace: 0
  showDecorationFinishCodeBlockTrace: 0
  endLogFileWith: '--------------'
  showGitHubInfoFooter: 1
  Dumper:
    Terse: 1
    Indent: 1
    Useqq: 1
    Deparse: 1
    Quotekeys: 0
    Sortkeys: 1
    Pair: " => "
```

When either of the `trace` modes (see page 17) are active, you will receive detailed information in `indent.log`. You can specify character strings to appear before and after the notification of a found code block using, respectively, `showDecorationStartCodeBlockTrace` and `showDecorationFinishCodeBlockTrace`. A demonstration is given in appendix I on page 177.
5.2 Verbatim code blocks

The log file will end with the characters given in endLogFileWith, and will report the GitHub address of latexindent.pl to the log file if showGitHubInfoFooter is set to 1.

Note: latexindent.pl no longer uses the log4perl module to handle the creation of the logfile.

Some of the options for Perl’s Dumper module can be specified in Listing 37; see [33] and [32] for more information. These options will mostly be helpful for those calling latexindent.pl with the \-tt option described in Section 3.2.

5.2 Verbatim code blocks

### verbatimEnvironments: (fields)

A field that contains a list of environments that you would like left completely alone – no indentation will be performed on environments that you have specified in this field, see Listing 38.

<table>
<thead>
<tr>
<th>Listing 38: verbatimEnvironments</th>
<th>Listing 39: verbatimCommands</th>
</tr>
</thead>
<tbody>
<tr>
<td>108 verbatimEnvironments:</td>
<td>114 verbatimCommands:</td>
</tr>
<tr>
<td>109     verbatim: 1</td>
<td>115     verb: 1</td>
</tr>
<tr>
<td>110     lstlisting: 1</td>
<td>116     lstinline: 1</td>
</tr>
<tr>
<td>111     minted: 1</td>
<td></td>
</tr>
</tbody>
</table>

Note that if you put an environment in verbatimEnvironments and in other fields such as lookForAlignDelims or noAdditionalIndent then latexindent.pl will always prioritize verbatimEnvironments.

You can, optionally, specify the verbatim field using the name field which takes a regular expression as its argument; thank you to [18] for contributing this feature.

example 10

For demonstration, then assuming that your file contains the environments latexcode, latexcode*, pythoncode and pythoncode*, then the listings given in Listings 40 and 41 are equivalent.

<table>
<thead>
<tr>
<th>Listing 40: nameAsRegex1.yaml</th>
<th>Listing 41: nameAsRegex2.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>verbatimEnvironments:</td>
<td>verbatimEnvironments:</td>
</tr>
<tr>
<td>108 latexcode: 1</td>
<td>114 nameAsRegex:</td>
</tr>
<tr>
<td>109 latexcode*: 1</td>
<td>115 name: ‘\w+code*?’</td>
</tr>
<tr>
<td>110 pythoncode: 1</td>
<td>116 lookForThis: 1</td>
</tr>
<tr>
<td>111 pythoncode*: 1</td>
<td></td>
</tr>
</tbody>
</table>

With reference to Listing 41:

- the name field as specified here means any word followed by the word code, optionally followed by *;
- we have used nameAsRegex to identify this field, but you can use any description you like;
- the lookForThis field is optional, and can take the values 0 (off) or 1 (on); by default, it is assumed to be 1 (on).

### verbatimCommands: (fields)

A field that contains a list of commands that are verbatim commands, for example \lstinline; any commands populated in this field are protected from line breaking routines (only relevant if the \-m is active, see Section 6 on page 81).

With reference to Listing 39, by default latexindent.pl looks for \verb immediately followed by another character, and then it takes the body as anything up to the next occurrence of the character; this means that, for example, \verb!x+3! is treated as a verbatimCommands.

You can, optionally, specify the verbatimCommands field using the name field which takes a regular expression as its argument; thank you to [18] for contributing this feature.
### 5.2 Verbatim code blocks

**Example 11**

For demonstration, then assuming that your file contains the commands `verbinline`, `myinline` then the listings given in Listings 42 and 43 are equivalent.

<table>
<thead>
<tr>
<th>Listing 42: <code>nameAsRegex3.yaml</code></th>
<th>Listing 43: <code>nameAsRegex4.yaml</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>verbatimCommands:</code></td>
<td><code>verbatimCommands:</code></td>
</tr>
<tr>
<td><code>verbinline: 1</code></td>
<td><code>nameAsRegex:</code></td>
</tr>
<tr>
<td><code>myinline: 1</code></td>
<td><code>name: \\w+inline</code></td>
</tr>
<tr>
<td></td>
<td><code>lookForThis: 1</code></td>
</tr>
</tbody>
</table>

With reference to Listing 43:

- the `name` field as specified here means *any word followed by the word inline*;
- we have used `nameAsRegex` to identify this field, but you can use any description you like;
- the `lookForThis` field is optional, and can take the values 0 (off) or 1 (on); by default, it is assumed to be 1 (on).

<table>
<thead>
<tr>
<th>noIndentBlock: (fields)</th>
</tr>
</thead>
</table>

If you have a block of code that you don’t want `latexindent.pl` to touch (even if it is *not* a verbatim-like environment) then you can wrap it in an environment from `noIndentBlock`; you can use any name you like for this, provided you populate it as demonstrate in Listing 44.

<table>
<thead>
<tr>
<th>Listing 44: <code>noIndentBlock</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>121  <code>noIndentBlock:</code></td>
</tr>
<tr>
<td>122  <code>noindent: 1</code></td>
</tr>
<tr>
<td>123  <code>cmhtest: 1</code></td>
</tr>
</tbody>
</table>

Of course, you don’t want to have to specify these as null environments in your code, so you use them with a comment symbol, `%`, followed by as many spaces (possibly none) as you like; see Listing 45 for example.

<table>
<thead>
<tr>
<th>Listing 45: <code>noIndentBlock.tex</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><code>% \begin{noindent}</code></td>
</tr>
<tr>
<td><code>some before text</code></td>
</tr>
<tr>
<td><code>this code won’t be touched</code></td>
</tr>
<tr>
<td><code>by latexindent.pl!</code></td>
</tr>
<tr>
<td><code>some after text</code></td>
</tr>
<tr>
<td><code>% \end{noindent}</code></td>
</tr>
</tbody>
</table>

Important note: it is assumed that the `noindent` block statements specified in this way appear on their own line.

**Example 12**

The `noIndentBlock` fields can also be specified in terms of `begin` and `end` fields. We use the code in Listing 46 to demonstrate this feature.

---

N: 2021-06-19
5.2 Verbatim code blocks

\begin{Verbatim}
\texttt{\textbackslash noIndentBlock1.tex}
\end{Verbatim}

some before text
\begin{verbatim}
this code
won't
be touched
\end{verbatim}
\begin{verbatim}
by
latexindent.pl!
\end{verbatim}
some after text

The settings given in Listings 47 and 48 are equivalent:

\begin{Verbatim}
\texttt{\textbackslash noindent1.yaml}
\end{Verbatim}

\begin{Verbatim}
\texttt{\textbackslash noindent2.yaml}
\end{Verbatim}

\begin{Verbatim}
\texttt{\textbackslash noindent3.yaml}
\end{Verbatim}

Upon running the commands

\begin{verbatim}
cmh:\$ latexindent.pl -l noindent1.yaml noindent1

\end{verbatim}

then we receive the output given in Listing 50.

\begin{Verbatim}
\texttt{\textbackslash noIndentBlock1.tex using Listing 47 or Listing 48}
\end{Verbatim}

some before text
\begin{verbatim}
this code
won't
be touched
\end{verbatim}
\begin{verbatim}
by
latexindent.pl!
\end{verbatim}
some after text

The begin, body and end fields for noIndentBlock are all regular expressions. If the body field is not specified, then it takes a default value of \texttt{.*?} which is written explicitly in Listing 47. In this context, we interpret \texttt{.*?} in words as the fewest number of characters (possibly none) until the `end' field is reached.

The lookForThis field is optional, and can take the values 0 (off) or 1 (on); by default, it is assumed to be 1 (on).

\begin{Verbatim}
\texttt{example 13}
\end{Verbatim}

Using Listing 49 demonstrates setting lookForThis to 0 (off); running the command

\begin{verbatim}
cmh:\$ latexindent.pl -l noindent3.yaml noindent1
\end{verbatim}

gives the output in Listing 51.

\section*{Example}

Using Listing 49 demonstrates setting lookForThis to 0 (off); running the command

\begin{verbatim}
cmh:\$ latexindent.pl -l noindent3.yaml noindent1
\end{verbatim}

gives the output in Listing 51.
some before text
this code
won’t
be touched
by
latexindent.pl!
some after text

We will demonstrate this feature later in the documentation in Listing 583.

You can, optionally, specify the noIndentBlock field using the name field which takes a regular expression as its argument; thank you to [18] for contributing this feature.

declaration 14

For demonstration, then assuming that your file contains the environments testnoindent, testnoindent*, then the listings given in Listings 52 and 53 are equivalent.

With reference to Listing 53:

- the name field as specified here means any word followed by the word noindent, optionally followed by *
- we have used nameAsRegex to identify this field, but you can use any description you like;
- the lookForThis field is optional, and can take the values 0 (off) or 1 (on); by default, it is assumed to be 1 (on).

5.3 filecontents and preamble

Before latexindent.pl determines the difference between preamble (if any) and the main document, it first searches for any of the environments specified in fileContentsEnvironments, see Listing 54. The behaviour of latexindent.pl on these environments is determined by their location (preamble or not), and the value indentPreamble, discussed next.

The preamble of a document can sometimes contain some trickier code for latexindent.pl to operate upon. By default, latexindent.pl won’t try to operate on the preamble (as indentPreamble is set to 0, by default), but if you’d like latexindent.pl to try then change indentPreamble to 1.
lookForPreamble: \{fields\}

Not all files contain preamble; for example, sty, cls and bib files typically do not. Referencing Listing 55, if you set, for example, .tex to 0, then regardless of the setting of the value of indentPreamble, preamble will not be assumed when operating upon .tex files.

<table>
<thead>
<tr>
<th>Listing 55: lookForPreamble</th>
</tr>
</thead>
</table>
| 135 | lookForPreamble:  
| 136 | .tex: 1  
| 137 | .sty: 0  
| 138 | .cls: 0  
| 139 | .bib: 0  
| 140 | STDIN: 1 |

preambleCommandsBeforeEnvironments: 0|1

Assuming that latexindent.pl is asked to operate upon the preamble of a document, when this switch is set to 0 then environment code blocks will be sought first, and then command code blocks. When this switch is set to 1, commands will be sought first. The example that first motivated this switch contained the code given in Listing 56.

<table>
<thead>
<tr>
<th>Listing 56: Motivating preambleCommandsBeforeEnvironments</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
</tr>
<tr>
<td>preheadhook=\begin{mdframed}{style=myframedstyle},</td>
</tr>
<tr>
<td>postfoothook=\end{mdframed},</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

5.4 Indentation and horizontal space

defaultIndent: \{horizontal space\}

This is the default indentation used in the absence of other details for the code block with which we are working. The default value is \t which means a tab; we will explore customisation beyond defaultIndent in Section 5.8 on page 56.

If you're interested in experimenting with latexindent.pl then you can remove all indentation by setting defaultIndent: "".

removeTrailingWhitespace: \{fields\}

Trailing white space can be removed both before and after processing the document, as detailed in Listing 57; each of the fields can take the values 0 or 1. See Listings 471 to 473 on page 119 for before and after results. Thanks to [3] for providing this feature.

<table>
<thead>
<tr>
<th>Listing 57: removeTrailingWhitespace</th>
</tr>
</thead>
<tbody>
<tr>
<td>153</td>
</tr>
<tr>
<td>154</td>
</tr>
<tr>
<td>155</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 58: removeTrailingWhitespace (alt)</th>
</tr>
</thead>
<tbody>
<tr>
<td>removeTrailingWhitespace: 1</td>
</tr>
</tbody>
</table>

You can specify removeTrailingWhitespace simply as 0 or 1, if you wish; in this case, latexindent.pl will set both beforeProcessing and afterProcessing to the value you specify; see Listing 58.
5.5 Aligning at delimiters

\texttt{lookForAlignDelims: \{fields\}}

This contains a list of code blocks that are operated upon in a special way by \texttt{latexindent.pl} (see Listing 59). In fact, the fields in \texttt{lookForAlignDelims} can actually take two different forms: the \textit{basic} version is shown in Listing 59 and the \textit{advanced} version in Listing 62; we will discuss each in turn.

\begin{Verbatim}
\begin{verbatim}
lookForAlignDelims:
  tabular: 1
  tabularx: 1
  longtable: 1
  array: 1
  matrix: 1
  ...
\end{verbatim}
\end{Verbatim}

Specifying code blocks in this field instructs \texttt{latexindent.pl} to try and align each column by its alignment delimiters. It does have some limitations (discussed further in Section 10), but in many cases it will produce results such as those in Listings 60 and 61; running the command

```
$ latexindent.pl tabular1.tex
```

gives the output given in Listing 61.

\begin{Verbatim}
\begin{tabular}{cccc}
  1 & 2 & 3 & 4 \\
  5 & & 6 & \\
\end{tabular}
\end{Verbatim}

If you find that \texttt{latexindent.pl} does not perform satisfactorily on such environments then you can set the relevant key to 0, for example \texttt{tabular: 0}; alternatively, if you just want to ignore specific instances of the environment, you could wrap them in something from \texttt{noIndentBlock} (see Listing 44 on page 30).

If, for example, you wish to remove the alignment of the ``` within a delimiter-aligned block, then the advanced form of \texttt{lookForAlignDelims} shown in Listing 62 is for you.
5.5 Aligning at delimiters

```
LISTING 62: lookForAlignDelims (advanced)

158 lookForAlignDelims:
159  tabular:
160  delims: 1
161  alignDoubleBackSlash: 1
162  spacesBeforeDoubleBackSlash: 1
163  multiColumnGrouping: 0
164  alignRowsWithoutMaxDelims: 1
165  spacesBeforeAmpersand: 1
166  spacesAfterAmpersand: 1
167  justification: left
168  alignFinalDoubleBackSlash: 0
169  dontMeasure: 0
170  delimiterRegEx: (?<!\)(&)
171  delimiterJustification: left
172  lookForChildCodeBlocks: 1
173  alignContentAfterDoubleBackSlash: 0
174  spacesAfterDoubleBackSlash: 1
175  tabularx:
176    delims: 1
177  longtable: 1
```

Note that you can use a mixture of the basic and advanced form: in Listing 62 `tabular` and `tabularx` are advanced and `longtable` is basic. When using the advanced form, each field should receive at least 1 sub-field, and can (but does not have to) receive any of the following fields:

- **delims**: binary switch (0 or 1) equivalent to simply specifying, for example, `tabular: 1` in the basic version shown in Listing 59. If `delims` is set to 0 then the align at ampersand routine will not be called for this code block (default: 1);
- **alignDoubleBackSlash**: binary switch (0 or 1) to determine if `\` should be aligned (default: 1);
- **spacesBeforeDoubleBackSlash**: optionally specifies the number (integer ≥ 0) of spaces to be inserted before `\` (default: 1);
- **multiColumnGrouping**: binary switch (0 or 1) that details if `latexindent.pl` should group columns above and below a `\multicolumn` command (default: 0);
- **alignRowsWithoutMaxDelims**: binary switch (0 or 1) that details if rows that do not contain the maximum number of delimiters should be formatted so as to have the ampersands aligned (default: 1);
- **spacesBeforeAmpersand**: optionally specifies the number (integer ≥ 0) of spaces to be placed before ampersands (default: 1);
- **spacesAfterAmpersand**: optionally specifies the number (integer ≥ 0) of spaces to be placed after ampersands (default: 1);
- **justification**: optionally specifies the justification of each cell as either `left` or `right` (default: left);
- **alignFinalDoubleBackSlash** optionally specifies if the final double backslash should be used for alignment (default: 0);
- **dontMeasure** optionally specifies if user-specified cells, rows or the largest entries should not be measured (default: 0);
- **delimiterRegEx** optionally specifies the pattern matching to be used for the alignment delimiter (default: `'(?<!\)(&)'`);
- **delimiterJustification** optionally specifies the justification for the alignment delimiters (default: left); note that this feature is only useful if you have delimiters of different lengths in the same column, discussed in Section 5.5.4;
5.5 Aligning at delimiters

• lookForChildCodeBlocks optionally instructs `latexindent.pl` to search for child code blocks or not (default: 1), discussed in Section 5.5.5;

• alignContentAfterDoubleBackSlash optionally instructs `latexindent.pl` to align content after double back slash (default: 0), discussed in Section 5.5.6;

• spacesAfterDoubleBackSlash optionally specifies the number (integer ≥ 0) of spaces to be placed after the double back slash when `alignContentAfterDoubleBackSlash` is active; demonstrated in Section 5.5.6.

example 15

We will explore most of these features using the file `tabular2.tex` in Listing 63 (which contains a `\multicolumn` command), and the YAML files in Listings 64 to 70; we will explore `alignFinalDoubleBackSlash` in Listing 91; the `dontMeasure` feature will be described in Section 5.5.3, and `delimiterRegEx` in Section 5.5.4.

```latex
\begin{tabular}{cccc}
  A & B & C & D \\
  AAA & BBB & CCC & DDD \\
  \multicolumn{2}{c}{first heading} & \multicolumn{2}{c}{second heading} \\
  one & two & three & four \\
  five & & six & \\
  seven & & \\
\end{tabular}
```

Listing 64: `tabular2.yaml`

`lookForAlignDelims:`

- `tabular:`
  - `multiColumnGrouping: 1`

Listing 65: `tabular3.yaml`

`lookForAlignDelims:`

- `tabular:`
  - `alignRowsWithoutMaxDelims: 0`

Listing 66: `tabular4.yaml`

`lookForAlignDelims:`

- `tabular:`
  - `spacesBeforeAmpersand: 4`

Listing 67: `tabular5.yaml`

`lookForAlignDelims:`

- `tabular:`
  - `spacesAfterAmpersand: 4`

Listing 68: `tabular6.yaml`

`lookForAlignDelims:`

- `tabular:`
  - `alignDoubleBackSlash: 0`

Listing 69: `tabular7.yaml`

`lookForAlignDelims:`

- `tabular:`
  - `spacesBeforeDoubleBackSlash: 0`

Listing 70: `tabular8.yaml`

`lookForAlignDelims:`

- `tabular:`
  - `justification: "right"`

On running the commands

```bash
cmh:~$ latexindent.pl tabular2.tex
cmh:~$ latexindent.pl tabular2.tex -l tabular2.yaml
cmh:~$ latexindent.pl tabular2.tex -l tabular3.yaml
cmh:~$ latexindent.pl tabular2.tex -l tabular4.yaml
cmh:~$ latexindent.pl tabular2.tex -l tabular4.yaml,tabular5.yaml
cmh:~$ latexindent.pl tabular2.tex -l tabular5.yaml,tabular6.yaml
cmh:~$ latexindent.pl tabular2.tex -l tabular6.yaml,tabular7.yaml
cmh:~$ latexindent.pl tabular2.tex -l tabular7.yaml,tabular8.yaml
```
we obtain the respective outputs given in Listings 71 to 78.

**Listing 71: tabular2.tex default output**

\begin{tabular}{cccc}
A & B & C & D \\
AAA & BBB & CCC & DDD \\
\multicolumn{2}{c}{first heading} & \multicolumn{2}{c}{second heading} \\
one & two & three & four \\
five & & six & \\
seven & & & \\
\end{tabular}

**Listing 72: tabular2.tex using Listing 64**

\begin{tabular}{cccc}
A & B & C & D \\
AAA & BBB & CCC & DDD \\
\multicolumn{2}{c}{first heading} & \multicolumn{2}{c}{second heading} \\
one & two & three & four \\
five & & six & \\
seven & & & \\
\end{tabular}

**Listing 73: tabular2.tex using Listing 65**

\begin{tabular}{cccc}
A & B & C & D \\
AAA & BBB & CCC & DDD \\
\multicolumn{2}{c}{first heading} & \multicolumn{2}{c}{second heading} \\
one & two & three & four \\
five & & six & \\
seven & & & \\
\end{tabular}

**Listing 74: tabular2.tex using Listings 64 and 66**

\begin{tabular}{cccc}
A & B & C & D \\
AAA & BBB & CCC & DDD \\
\multicolumn{2}{c}{first heading} & \multicolumn{2}{c}{second heading} \\
one & two & three & four \\
five & & six & \\
seven & & & \\
\end{tabular}

**Listing 75: tabular2.tex using Listings 64 and 67**

\begin{tabular}{cccc}
A & B & C & D \\
AAA & BBB & CCC & DDD \\
\multicolumn{2}{c}{first heading} & \multicolumn{2}{c}{second heading} \\
one & two & three & four \\
five & & six & \\
seven & & & \\
\end{tabular}
5.5 Aligning at delimiters

LISTING 76: tabular2.tex using Listings 64 and 68

\begin{tabular}{cccc}
A & B & C & D \\
AAA & BBB & CCC & DDD \\
\multicolumn{2}{c}{first heading} & \multicolumn{2}{c}{second heading} \\
one & two & three & four \\
five & & six & \\
seven & \\
\end{tabular}

LISTING 77: tabular2.tex using Listings 64 and 69

\begin{tabular}{cccc}
A & B & C & D \\
AAA & BBB & CCC & DDD \\
\multicolumn{2}{c}{first heading} & \multicolumn{2}{c}{second heading} \\
one & two & three & four \\
five & & six & \\
seven & \\
\end{tabular}

LISTING 78: tabular2.tex using Listings 64 and 70

\begin{tabular}{cccc}
A & B & C & D \\
AAA & BBB & CCC & DDD \\
\multicolumn{2}{c}{first heading} & \multicolumn{2}{c}{second heading} \\
one & two & three & four \\
five & & six & \\
seven & \\
\end{tabular}

Notice in particular:

- in both Listings 71 and 72 all rows have been aligned at the ampersand, even those that do not contain the maximum number of ampersands (3 ampersands, in this case);
- in Listing 71 the columns have been aligned at the ampersand;
- in Listing 72 the \multicolumn command has grouped the 2 columns beneath and above it, because multiColumnGrouping is set to 1 in Listing 64;
- in Listing 73 rows 3 and 6 have not been aligned at the ampersand, because alignRowsWithoutMaxDelims has been set to 0 in Listing 65; however, the \ have still been aligned;
- in Listing 74 the columns beneath and above the \multicolumn statements have been grouped (because multiColumnGrouping is set to 1), and there are at least 4 spaces before each aligned ampersand because spacesBeforeAmpersand is set to 4;
- in Listing 75 the columns beneath and above the \multicolumn commands have been grouped (because multiColumnGrouping is set to 1), and there are at least 4 spaces after each aligned ampersand because spacesAfterAmpersand is set to 4;
- in Listing 76 the \ have not been aligned, because alignDoubleBackSlash is set to 0, otherwise the output is the same as Listing 72;
- in Listing 77 the \ have been aligned, and because spacesBeforeDoubleBackSlash is set to 0, there are no spaces ahead of them; the output is otherwise the same as Listing 72;
- in Listing 78 the cells have been right-justified; note that cells above and below the \multicol statements have still been group correctly, because of the settings in Listing 64.
5.5 Aligning at delimiters

5.5.1 lookForAlignDelims: spacesBeforeAmpersand

The spacesBeforeAmpersand can be specified in a few different ways. The basic form is demonstrated in Listing 66, but we can customise the behaviour further by specifying if we would like this value to change if it encounters a leading blank column; that is, when the first column contains only zero-width entries. We refer to this as the advanced form.

example 16

We demonstrate this feature in relation to Listing 79; upon running the following command

```
cmh:~$ latexindent.pl aligned1.tex -o=+-default
```

then we receive the default output given in Listing 80.

<table>
<thead>
<tr>
<th>Listing 79: aligned1.tex</th>
<th>Listing 80: aligned1-default.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{aligned}</td>
<td>\begin{aligned}</td>
</tr>
<tr>
<td>&amp; a &amp; b, \</td>
<td>&amp; a &amp; b, \</td>
</tr>
<tr>
<td>&amp; c &amp; d.</td>
<td>&amp; c &amp; d.</td>
</tr>
<tr>
<td>\end{aligned}</td>
<td>\end{aligned}</td>
</tr>
</tbody>
</table>

The settings in Listings 81 to 84 are all equivalent; we have used the not-yet discussed noAdditionalIndent field (see Section 5.8 on page 56) which will assist in the demonstration in what follows.

<table>
<thead>
<tr>
<th>Listing 81: sba1.yaml</th>
<th>Listing 82: sba2.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>noAdditionalIndent:</td>
<td>noAdditionalIndent:</td>
</tr>
<tr>
<td>aligned: 1</td>
<td>aligned: 1</td>
</tr>
<tr>
<td>lookForAlignDelims:</td>
<td>lookForAlignDelims:</td>
</tr>
<tr>
<td>aligned: 1</td>
<td>aligned:</td>
</tr>
<tr>
<td></td>
<td>spacesBeforeAmpersand:</td>
</tr>
<tr>
<td></td>
<td>default: 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 83: sba3.yaml</th>
<th>Listing 84: sba4.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>noAdditionalIndent:</td>
<td>noAdditionalIndent:</td>
</tr>
<tr>
<td>aligned: 1</td>
<td>aligned: 1</td>
</tr>
<tr>
<td>lookForAlignDelims:</td>
<td>lookForAlignDelims:</td>
</tr>
<tr>
<td>aligned:</td>
<td>aligned:</td>
</tr>
<tr>
<td>spacesBeforeAmpersand:</td>
<td>spacesBeforeAmpersand:</td>
</tr>
<tr>
<td>default: 1</td>
<td>leadingBlankColumn: 1</td>
</tr>
</tbody>
</table>

Upon running the following commands

```
cmh:~$ latexindent.pl aligned1.tex -l sba1.yaml
```
```
cmh:~$ latexindent.pl aligned1.tex -l sba2.yaml
```
```
cmh:~$ latexindent.pl aligned1.tex -l sba3.yaml
```
```
cmh:~$ latexindent.pl aligned1.tex -l sba4.yaml
```

then we receive the (same) output given in Listing 85; we note that there is one space before each ampersand.

<table>
<thead>
<tr>
<th>Listing 85: aligned1-mod1.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{aligned}</td>
</tr>
<tr>
<td>&amp; a &amp; b, \</td>
</tr>
<tr>
<td>&amp; c &amp; d.</td>
</tr>
<tr>
<td>\end{aligned}</td>
</tr>
</tbody>
</table>

We note in particular:

- Listing 81 demonstrates the basic form for lookForAlignDelims; in this case, the default values are specified as in Listing 62 on page 35;
- Listing 82 demonstrates the advanced form for lookForAlignDelims and specified spacesBeforeAmpersand.
5.5 Aligning at delimiters

The default value is 1;

- Listing 83 demonstrates the new advanced way to specify `spacesBeforeAmpersand`, and for us to set the default value that sets the number of spaces before ampersands which are not in leading blank columns. The default value is 1.

We note that `leadingBlankColumn` has not been specified in Listing 83, and it will inherit the value from default;

- Listing 84 demonstrates spaces to be used before amperands for leading blank columns. We note that `default` has not been specified, and it will be set to 1 by default.

example 17

We can customise the space before the ampersand in the leading blank column of Listing 85 by using either of Listings 86 and 87, which are equivalent.

<table>
<thead>
<tr>
<th>Listing 86: sba5.yaml</th>
<th>Listing 87: sba6.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>noAdditionalIndent:</td>
<td>noAdditionalIndent:</td>
</tr>
<tr>
<td>aligned: 1</td>
<td>aligned: 1</td>
</tr>
<tr>
<td>lookForAlignDelims:</td>
<td>lookForAlignDelims:</td>
</tr>
<tr>
<td>aligned:</td>
<td>aligned:</td>
</tr>
<tr>
<td>spacesBeforeAmpersand:</td>
<td>spacesBeforeAmpersand:</td>
</tr>
<tr>
<td>0</td>
<td>leadingBlankColumn: 0</td>
</tr>
<tr>
<td>default:</td>
<td>default: 1</td>
</tr>
</tbody>
</table>

Upon running

```
cmh:~$ latexindent.pl aligned1.tex -l sba5.yaml
```

then we receive the (same) output given in Listing 88. We note that the space before the ampersand in the leading blank column has been set to 0 by Listing 87.

We can demonstrated this feature further using the settings in Listing 90 which give the output in Listing 89.

<table>
<thead>
<tr>
<th>Listing 88: aligned1-mod5.tex</th>
<th>Listing 89: aligned1.tex using Listing 90</th>
<th>Listing 90: sba7.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{aligned} &amp; a &amp; b, \ &amp; c &amp; d. \end{aligned}</td>
<td>\begin{aligned} &amp; a &amp; b, \ &amp; c &amp; d. \end{aligned}</td>
<td>\begin{aligned} &amp; a &amp; b, \ &amp; c &amp; d. \end{aligned}</td>
</tr>
<tr>
<td></td>
<td></td>
<td>noAdditionalIndent:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>aligned: 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>lookForAlignDelims:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>aligned:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>spacesBeforeAmpersand:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>leadingBlankColumn: 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>default: 0</td>
</tr>
</tbody>
</table>

5.5.2 lookForAlignDelims: `alignFinalDoubleBackSlash`

There may be times when a line of a code block contains more than `\\`, and in which case, you may want the final double backslash to be aligned.

example 18

We explore the `alignFinalDoubleBackSlash` feature by using the file in Listing 91. Upon running the following commands

```
cmh:~$ latexindent.pl tabular4.tex -o++-default
```

N: 2020-03-21
then we receive the respective outputs given in Listing 92 and Listing 93.

<table>
<thead>
<tr>
<th>Listing 91: tabular4.tex</th>
<th>Listing 92: tabular4-default.tex</th>
<th>Listing 93: tabular4-FDBS.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{tabular}{lc}</td>
<td>\begin{tabular}{lc}</td>
<td>\begin{tabular}{lc}</td>
</tr>
<tr>
<td>\textbf{Name} &amp; \shortstack{Hi \ Lo} \ \</td>
<td>\textbf{Name} &amp; \shortstack{Hi \ Lo} \ \</td>
<td>\textbf{Name} &amp; \shortstack{Hi \ Lo} \ \</td>
</tr>
<tr>
<td>\textbf{Foo} &amp; \textbf{Bar} \ \</td>
<td>\textbf{Foo} &amp; \textbf{Bar} \ \</td>
<td>\textbf{Foo} &amp; \textbf{Bar} \ \</td>
</tr>
<tr>
<td>\end{tabular}</td>
<td>\end{tabular}</td>
<td>\end{tabular}</td>
</tr>
</tbody>
</table>

We note that in:

- Listing 92, by default, the first set of double back slashes in the first row of the tabular environment have been used for alignment;

- Listing 93, the final set of double backslashes in the first row have been used, because we specified alignFinalDoubleBackSlash as 1.

As of Version 3.0, the alignment routine works on mandatory and optional arguments within commands, and also within ‘special’ code blocks (see specialBeginEnd on page 48).

example 19

Assuming that you have a command called \matrix and that it is populated within lookForAlignDelims (which it is, by default), and that you run the command

```
\texttt{cmh:~$ \texttt{latexindent.pl matrix1.tex}}
```

then the before-and-after results shown in Listings 94 and 95 are achievable by default.

<table>
<thead>
<tr>
<th>Listing 94: matrix1.tex</th>
<th>Listing 95: matrix1.tex default output</th>
</tr>
</thead>
<tbody>
<tr>
<td>\matrix [</td>
<td>\matrix [</td>
</tr>
<tr>
<td>1 &amp; 2 &amp; 3 \ \</td>
<td>1 &amp; 2 &amp; 3 \ \</td>
</tr>
<tr>
<td>4 &amp; 5 &amp; 6]</td>
<td>4 &amp; 5 &amp; 6]</td>
</tr>
<tr>
<td>7 &amp; 8 &amp; 9 \ \</td>
<td>7 &amp; 8 &amp; 9 \ \</td>
</tr>
<tr>
<td>10 &amp; 11 &amp; 12</td>
<td>10 &amp; 11 &amp; 12</td>
</tr>
<tr>
<td>]</td>
<td>]</td>
</tr>
</tbody>
</table>

If you have blocks of code that you wish to align at the & character that are not wrapped in, for example, \begin{tabular}...\end{tabular}, then you can use the mark up illustrated in Listing 96; the default output is shown in Listing 97. Note that the %* must be next to each other, but that there can be any number of spaces (possibly none) between the * and \begin{tabular}; note also that you may use any environment name that you have specified in lookForAlignDelims.

<table>
<thead>
<tr>
<th>Listing 96: align-block.tex</th>
<th>Listing 97: align-block.tex default output</th>
</tr>
</thead>
<tbody>
<tr>
<td>%* \begin{tabular}</td>
<td>%* \begin{tabular}</td>
</tr>
<tr>
<td>1 &amp; 2 &amp; 3 &amp; 4 \ \</td>
<td>1 &amp; 2 &amp; 3 &amp; 4 \ \</td>
</tr>
<tr>
<td>5 &amp; \ \ \ \</td>
<td>5 &amp; \ \ \ \</td>
</tr>
<tr>
<td>%* \end{tabular}</td>
<td>%* \end{tabular}</td>
</tr>
</tbody>
</table>

With reference to Table 2 on page 57 and the, yet undiscussed, fields of noAdditionalIndent and indentRules (see Section 5.8 on page 56), these comment-marked blocks are considered environments.

### 5.5.3 lookForAlignDelims: the dontMeasure feature

The lookForAlignDelims field can, optionally, receive the dontMeasure option which can be specified in a few different ways.

**example 20**

We will explore this feature in relation to the code given in Listing 98; the default output is shown in Listing 99.
The `dontMeasure` field can be specified as `largest`, and in which case, the largest element will not be measured; with reference to the YAML file given in Listing 101, we can run the command

```
cmh::∼$ latexindent.pl tabular-DM.tex -l=dontMeasure1.yaml
```

and receive the output given in Listing 100.

```
\begin{tabular}{cccc}
  aaaaaa & bbbbb & ccc & dd \\
  11 & 2 & 33 & 4 \\
  5 & 66 & 7 & 8
\end{tabular}
```

We note that the `largest` column entries have not contributed to the measuring routine.

**Example 21**

The `dontMeasure` field can also be specified in the form demonstrated in Listing 103. On running the following commands,

```
cmh::∼$ latexindent.pl tabular-DM.tex -l=dontMeasure2.yaml
```

we receive the output in Listing 102.

```
\begin{tabular}{cccc}
  aaaaaa & bbbbb & ccc & dd \\
  11 & 2 & 33 & 4 \\
  5 & 66 & 7 & 8
\end{tabular}
```

We note that in Listing 103 we have specified entries not to be measured, one entry per line.

**Example 22**

The `dontMeasure` field can also be specified in the forms demonstrated in Listing 105 and Listing 106. Upon running the commands

```
cmh::∼$ latexindent.pl tabular-DM.tex -l=dontMeasure3.yaml

cmh::∼$ latexindent.pl tabular-DM.tex -l=dontMeasure4.yaml
```

we receive the output given in Listing 104.
5.5 Aligning at delimiters

We note that in:

- Listing 105 we have specified entries not to be measured, each one has a string in the `this` field, together with an optional specification of `applyTo` as `cell`;
- Listing 106 we have specified entries not to be measured as a regular expression using the `regex` field, together with an optional specification of `applyTo` as `cell` field, together with an optional specification of `applyTo` as `cell`.

In both cases, the default value of `applyTo` is `cell`, and does not need to be specified.

**example 23**

We may also specify the `applyTo` field as `row`, a demonstration of which is given in Listing 108; upon running

```
$ latexindent.pl tabular-DM.tex -l=dontMeasure5.yaml
```

we receive the output in Listing 107.

**example 24**

Finally, the `applyTo` field can be specified as `row`, together with a regex expression. For example, for the settings given in Listing 110, upon running

```
$ latexindent.pl tabular-DM.tex -l=dontMeasure6.yaml
```

we receive the output in Listing 109.
5.5 Aligning at delimiters

5.5.4 lookForAlignDelims: the delimiterRegEx and delimiterJustification feature

The delimiter alignment will, by default, align code blocks at the ampersand character. The behaviour is controlled by the delimiterRegEx field within lookForAlignDelims; the default value is '\(?<!\\)(=|>)\)', which can be read as: an ampersand, as long as it is not immediately preceded by a backslash.

**Warning!**

Important: note the ‘capturing’ parenthesis in the (>) which are necessary; if you intend to customise this field, then be sure to include them appropriately.

**example 25**

We demonstrate how to customise this with respect to the code given in Listing 111; the default output from latexindent.pl is given in Listing 112.

We note that:

- in Listing 113 the code has been aligned, as intended, at both the \= or \>.

**example 26**

We can explore delimiterRegEx a little further using the settings in Listing 116 and run the command

cmh:~$ latexindent.pl tabbing.tex -l=delimiterRegEx2.yaml

to receive the output given in Listing 115.
We note that only the \> have been aligned.

example 27

Of course, the other lookForAlignDelims options can be used alongside the delimiterRegEx; regardless of the type of delimiter being used (ampersand or anything else), the fields from Listing 62 on page 35 remain the same; for example, using the settings in Listing 118, and running

```
cmh:~$ latexindent.pl tabbing.tex -l=delimiterRegEx3.yaml
cmh:~$ latexindent.pl tabbing1.tex -l=delimiterRegEx4.yaml -o=+-mod4
```

to receive the output given in Listing 117.

example 28

It is possible that delimiters specified within delimiterRegEx can be of different lengths. Consider the file in Listing 119, and associated YAML in Listing 121. Note that the Listing 121 specifies the option for the delimiter to be either # or \>, which are different lengths. Upon running the command

```
cmh:~$ latexindent.pl tabbing1.tex -l=delimiterRegEx4.yaml -o=+-mod4
```

we receive the output in Listing 120.

```
\begin{tabbing}
1 # 22 \> 333 \\\nxxx # aaa # yyyyy \\\n\. # # & \\
\end{tabbing}
```

example 29

You can set the delimiter justification as either left (default) or right, which will only have effect when delimiters in the same column have different lengths. Using the settings in Listing 123 and running the command

```
cmh:~$ latexindent.pl tabbing1.tex -l=delimiterRegEx5.yaml -o=+-mod5
```
gives the output in Listing 122.

Note that in Listing 122 the second set of delimiters have been right aligned – it is quite subtle!

5.5.5 **lookForAlignDelims: lookForChildCodeBlocks**

There may be scenarios in which you would prefer to instruct latexindent.pl not to search for child blocks; in which case setting lookForChildCodeBlocks to 0 may be a good way to proceed.

**example 30**

Using the settings from Listing 101 on page 42 on the file in Listing 124 and running the command

```bash
cmh:~$ latexindent.pl tabular-DM-1.tex -l=dontMeasure1.yaml -o=-mod1
```

gives the output in Listing 125.

We can improve the output from Listing 125 by employing the settings in Listing 127

```bash
cmh:~$ latexindent.pl tabular-DM-1.tex -l=dontMeasure1a.yaml -o=-mod1a
```

which gives the output in Listing 127.

5.5.6 **lookForAlignDelims: alignContentAfterDoubleBackSlash**

You can instruct latexindent to align content after the double back slash. See also Section 6.3.2 on page 121.

**example 31**

We consider the file in Listing 128, and the default output given in Listing 129.

Using the settings given in Listing 131 and running
5.6 Indent after items, specials and headings

\begin{itemize}
\item some text here
\item some more text here
\item another item
some more text here
\end{itemize}

\begin{itemize}
\item some text here
\item some more text here
\item another item
some more text here
\end{itemize}

If you have your own item commands (perhaps you prefer to use myitem, for example) then you can put populate them in itemNames. For example, users of the exam document class might like to add parts to indentAfterItems and part to itemNames to their user settings (see Section 4 on page 23 for details of how to configure user settings, and Listing 33 on page 24 in particular.)
5.6 Indent after items, specials and headings

The fields specified in `specialBeginEnd` are, in their default state, focused on math mode begin and end statements, but there is no requirement for this to be the case; Listing 138 shows the default settings of `specialBeginEnd`.

```
L\ISTING 138: specialBeginEnd
specialBeginEnd:
  displayMath:
    begin: (?:!\\)[ # \[ but *not* \[\]
    end: \\\] # \]
    lookForThis: 1
  inlineMath:
    begin: (?<!\$)(?<!\\)$?(?!\$) # $ but *not* $ or $$
    body: \[^$]*? # anything *except* $
    end: (?<!\$)$?(?!\$) # $ but *not* $ or $$
    lookForThis: 1
  displayMathTeX:
    begin: $\$
    end: $\$ # $$
    lookForThis: 1
  specialBeforeCommand: 0
```

The field `displayMath` represents \[\ldots\], `inlineMath` represents $\ldots$ and `displayMathTeX` represents $$\ldots$$. You can, of course, rename these in your own YAML files (see Section 4.2 on page 24); indeed, you might like to set up your own special begin and end statements.

example 33

A demonstration of the before-and-after results are shown in Listings 139 and 140; explicitly, running the command

```
cmh:~$ latexindent.pl special1.tex -o=+-default
```
gives the output given in Listing 140.

```
L\ISTING 139: special1.tex before
The function $f$ has formula
\[
  f(x)=x^2.
\]
If you like splitting dollars,
$
  g(x)=f(2x)
$
```

```
L\ISTING 140: special1.tex default output
The function $f$ has formula
\[
  f(x)=x^2.
\]
If you like splitting dollars,
$
  g(x)=f(2x)
$
```

For each field, `lookForThis` is set to 1 by default, which means that `latexindent.pl` will look for this pattern; you can tell `latexindent.pl` not to look for the pattern, by setting `lookForThis` to 0.

There are examples in which it is advantageous to search for `specialBeginEnd` fields before searching for commands, and the `specialBeforeCommand` switch controls this behaviour.
example 34

For example, consider the file shown in Listing 141.

<table>
<thead>
<tr>
<th>LISTING 141: specialLR.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{equation} \left[ \sqrt{a+b} \right] \end{equation}</td>
</tr>
</tbody>
</table>

Now consider the YAML files shown in Listings 142 and 143

<table>
<thead>
<tr>
<th>LISTING 142: specialsLeftRight.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>specialBeginEnd:</td>
</tr>
<tr>
<td>leftRightSquare:</td>
</tr>
<tr>
<td>begin: '\left[',</td>
</tr>
<tr>
<td>end: '\right]',</td>
</tr>
<tr>
<td>lookForThis: 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LISTING 143: specialBeforeCommand.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>specialBeginEnd:</td>
</tr>
<tr>
<td>specialBeforeCommand: 1</td>
</tr>
</tbody>
</table>

Upon running the following commands

cmh:$ \texttt{latexindent.pl specialLR.tex -l=specialsLeftRight.yaml}
cmh:$ \texttt{latexindent.pl specialLR.tex -l=specialsLeftRight.yaml,specialBeforeCommand.yaml}

we receive the respective outputs in Listings 144 and 145.

<table>
<thead>
<tr>
<th>LISTING 144: specialLR.tex using Listing 142</th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{equation} \left[ \sqrt{a+b} \right] \end{equation}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LISTING 145: specialLR.tex using Listings 142 and 143</th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{equation} \left[ \sqrt{a+b} \right] \end{equation}</td>
</tr>
</tbody>
</table>

Notice that in:

- Listing 144 the \texttt{\left} has been treated as a \texttt{command}, with one optional argument;
- Listing 145 the \texttt{specialBeginEnd} pattern in Listing 142 has been obeyed because Listing 143 specifies that the \texttt{specialBeginEnd} should be sought \texttt{before} commands.

You can, optionally, specify the \texttt{middle} field for anything that you specify in \texttt{specialBeginEnd}.

example 35

For example, let’s consider the .tex file in Listing 146.
5.6 Indent after items, specials and headings

Upon saving the YAML settings in Listings 148 and 150 and running the commands

```bash
cmh:~$ latexindent.pl special2.tex -l=middle
```

then we obtain the output given in Listings 147 and 149.

We note that:

- in Listing 147 the bodies of each of the Elsif statements have been indented appropriately;
- the Else statement has not been indented appropriately in Listing 147 – read on!
- we have specified multiple settings for the middle field using the syntax demonstrated in Listing 150 so that the body of the Else statement has been indented appropriately in Listing 149.

You may need these fields in your own YAML files (see Section 4.2 on page 24), if you use popular algorithm packages such as algorithms, algorithm2e or algpseudocode, etc.
For example, let's consider the .tex file in Listing 151.

**Listing 151: specialAlgo.tex**

```latex
\begin{verbatim}
\SetNoFillState
\State body
\EndFor
\EndFor
\EndFor
$[\text{equality}\ if\ gdc\ is\ 1]\}$\Comment\{The g.c.d. of a and b\}
\While[($[\text{var}\ =\ end]\ )\Comment\{We\ have\ the\ answer\ if\ \text{r}\ \ is\ 0\}]
\State $[\text{var}\gets\text{var}\ mod\ \text{b}]$
\EndWhile
\State $[\text{value}\{\text{return}\}\ \text{value}\{\text{The\ gcd\ is\ } b\}]$
\EndFunction
\end{verbatim}
```

Upon saving the YAML settings in Listing 153 and running the command

```
\texttt{cmh:~}\$ latexindent.pl -l=algo.yaml specialAlgo.tex
```

then we obtain the output given in Listing 152.

**Listing 152: specialAlgo.tex using Listing 153**

```yaml
specialBeginEnd:
  Procedure:
    begin: \{\For\{[^}\]+?\}\}\{[^}\]+?\}\end{verbatim}
\State body
\EndLoops
\State $[\text{var}\gets\text{end}]$
\Repeat
\State $[\text{var}\gets\text{end}]$\{\For\{[^}\]+?\}\}\{[^}\]+?\}\end{verbatim}
\EndWhile
\State $[\text{value}\{\text{return}\}\ \text{value}\{\text{The\ gcd\ is\ } b\}]$
\EndFunction
```

You may specify fields in specialBeginEnd to be treated as verbatim code blocks by changing lookForThis to be verbatim.
5.6 Indent after items, specials and headings

example 37

For example, beginning with the code in Listing 154 and the YAML in Listing 155, and running

```
$ latexindent.pl special3.tex -l=instrument.yaml
```

then the output in Listing 154 is unchanged.

```
\begin{tikzpicture}
\path (A) edge node {0,1,L} (B)
    edge node {1,1,R} (C)
    edge [loop above] node {1,1,L} (B)
    edge node {0,1,L} (C)
    edge [bend left] node {1,0,R} (E)
    edge [loop below] node {1,1,R} (D)
    edge node {0,1,R} (A)
    edge [bend left] node {1,0,R} (A);
\end{tikzpicture}
```

We can combine the `specialBeginEnd` with the `lookForAlignDelims` feature.

example 38

We begin with the code in Listing 156.

```
\begin{tikzpicture}
\path (A) edge node {0,1,L} (B)
    edge node {1,1,R} (C)
    edge [loop above] node {1,1,L} (B)
    edge node {0,1,L} (C)
    edge [bend left] node {1,0,R} (E)
    edge [loop below] node {1,1,R} (D)
    edge node {0,1,R} (A)
    edge [bend left] node {1,0,R} (A);
\end{tikzpicture}
```

Let’s assume that our goal is to align the code at the `edge` and `node` text; we employ the code given in Listing 158 and run the command

```
$ latexindent.pl special-align.tex -l edge-node1.yaml -o=-mod1
```

to receive the output in Listing 157.

The output in Listing 157 is not quite ideal. We can tweak the settings within Listing 158 in order to improve the output; in particular, we employ the code in Listing 160 and run the command
5.6 Indent after items, specials and headings

```bash
cmh:~$ latexindent.pl special-align.tex -l edge-node2.yaml -o=+-mod2
```

to receive the output in Listing 159.

<table>
<thead>
<tr>
<th>LISTING 159: special-align.tex using Listing 160</th>
</tr>
</thead>
</table>

```latex
\begin{tikzpicture}
\path (A) edge node {0,1,L} (B) edge node {1,1,R} (C) 
(B) edge [loop above] node {1,1,L} (B) edge node {0,1,L} (C) 
(C) edge node {0,1,L} (D) edge node {1,0,R} (E) 
(D) edge [loop below] node {1,1,R} (D) edge node {0,1,R} (A) 
(E) edge [bend left] node {1,0,R} (A);
\end{tikzpicture}
```

<table>
<thead>
<tr>
<th>LISTING 160: edge-node2.yaml</th>
</tr>
</thead>
</table>

```yaml
specialBeginEnd:
  path:
    begin: '\path'
    end: ' ;;'
  specialBeforeCommand: 1

lookForAlignDelims:
  path:
    delimiterRegEx: '(edge|node\{[0-9,A-Z]+\})'
```

The `lookForThis` field can be considered optional; by default, it is assumed to be 1, which is demonstrated in Listing 160.

Referencing Listing 138 on page 48 we see that each of the `specialBeginEnd` fields can optionally accept the body field. If the body field is omitted, then `latexindent.pl` uses a value that means anything except one of the begin statements from `specialBeginEnd`.

In general, it is usually _not_ necessary to specify the body field, but let’s detail an example just for reference.

**example 39**

We begin with the example in Listing 161

```
\begin{tikzpicture}
\path (A) edge node {0,1,L} (B) edge node {1,1,R} (C)
(B) edge [loop above] node {1,1,L} (B) edge node {0,1,L} (C)
(C) edge node {0,1,L} (D) edge node {1,0,R} (E)
(D) edge [loop below] node {1,1,R} (D) edge node {0,1,R} (A)
(E) edge [bend left] node {1,0,R} (A);
\end{tikzpicture}
```

Using the settings in Listing 163 and running the command

```bash
cmh:~$ latexindent.pl special-body.tex -l=special-body1.yaml
```
gives the output in Listing 162.
5.6 Indent after items, specials and headings

We note that the output in Listing 162 is as we would expect, even without the body field specified. Another option (purely for reference) that leaves the output in Listing 162 unchanged is shown in Listing 164.

The body field in Listing 164 means anything except ( or ).

This field enables the user to specify indentation rules that take effect after heading commands such as \part, \chapter, \section, \subsection*, or indeed any user-specified command written in this field.¹

The default settings do not place indentation after a heading, but you can easily switch them on by changing indentAfterThisHeading from 0 to 1. The level field tells latexindent.pl the hierarchy of the heading structure in your document. You might, for example, like to have both section and subsection set with level: 3 because you do not want the indentation to go too

¹There is a slight difference in interface for this field when comparing Version 2.2 to Version 3.0; see appendix L on page 180 for details.
5.6 Indent after items, specials and headings

You can add any of your own custom heading commands to this field, specifying the level as appropriate. You can also specify your own indentation in indentRules (see Section 5.8 on the next page); you will find the default indentRules contains chapter: “ ” which tells latexindent.pl simply to use a space character after chapter headings (once indent is set to 1 for chapter).

example 40

For example, assuming that you have the code in Listing 167 saved into headings1.yaml, and that you have the text from Listing 166 saved into headings1.tex.

<table>
<thead>
<tr>
<th>LISTING 166: headings1.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>\subsection{subsection title}</td>
</tr>
<tr>
<td>subsection text</td>
</tr>
<tr>
<td>subsection text</td>
</tr>
<tr>
<td>\paragraph{paragraph title}</td>
</tr>
<tr>
<td>paragraph text</td>
</tr>
<tr>
<td>paragraph text</td>
</tr>
<tr>
<td>\paragraph{paragraph title}</td>
</tr>
<tr>
<td>paragraph text</td>
</tr>
<tr>
<td>paragraph text</td>
</tr>
</tbody>
</table>

If you run the command

cmh:~$ latexindent.pl headings1.tex -l=headings1.yaml

then you should receive the output given in Listing 168.

<table>
<thead>
<tr>
<th>LISTING 168: headings1.tex using Listing 167</th>
</tr>
</thead>
<tbody>
<tr>
<td>\subsection{subsection title}</td>
</tr>
<tr>
<td>\hspace*{1em}subsection text</td>
</tr>
<tr>
<td>\hspace*{1em}subsection text</td>
</tr>
<tr>
<td>\hspace*{2em}paragraph{paragraph title}</td>
</tr>
<tr>
<td>\hspace*{2em}paragraph text</td>
</tr>
<tr>
<td>\hspace*{2em}paragraph text</td>
</tr>
<tr>
<td>\hspace*{2em}paragraph{paragraph title}</td>
</tr>
<tr>
<td>\hspace*{2em}paragraph text</td>
</tr>
<tr>
<td>\hspace*{2em}paragraph text</td>
</tr>
</tbody>
</table>

Now say that you modify the YAML from Listing 167 so that the paragraph level is 1; after running

<table>
<thead>
<tr>
<th>LISTING 169: headings1.tex second modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>\subsection{subsection title}</td>
</tr>
<tr>
<td>\hspace*{1em}subsection text</td>
</tr>
<tr>
<td>\hspace*{1em}subsection text</td>
</tr>
<tr>
<td>\hspace*{2em}paragraph{paragraph title}</td>
</tr>
<tr>
<td>\hspace*{2em}paragraph text</td>
</tr>
<tr>
<td>\hspace*{2em}paragraph text</td>
</tr>
<tr>
<td>\hspace*{2em}paragraph{paragraph title}</td>
</tr>
<tr>
<td>\hspace*{2em}paragraph text</td>
</tr>
<tr>
<td>\hspace*{2em}paragraph text</td>
</tr>
</tbody>
</table>

you should receive the code given in Listing 169; notice that the paragraph and subsection are at the same indentation level.

maximumIndentation: \{horizontal space\}

You can control the maximum indentation given to your file by specifying the maximumIndentation field as horizontal space (but not including tabs). This feature uses the Text::Tabs module [46], and is off by default.

example 41

For example, consider the example shown in Listing 170 together with the default output shown in Listing 171.
5.7 The code blocks known latexindent.pl

As of Version 3.0, latexindent.pl processes documents using code blocks; each of these are shown in Table 2.

We will refer to these code blocks in what follows. Note that the fine tuning of the definition of the code blocks detailed in Table 2 is discussed in Section 9 on page 147.

5.8 noAdditionalIndent and indentRules

latexindent.pl operates on files by looking for code blocks, as detailed in Section 5.7; for each type of code block in Table 2 on the next page (which we will call a (thing) in what follows) it searches YAML fields for information in the following order:

1. noAdditionalIndent for the name of the current (thing);

example 42

Now say that, for example, you have the max-indentation1.yaml from Listing 173 and that you run the following command:

```
cmb:~$ latexindent.pl mult-nested.tex -l=max-indentation1
```

You should receive the output shown in Listing 172.

Comparing the output in Listings 171 and 172 we notice that the (default) tabs of indentation have been replaced by a single space.

In general, when using the maximumIndentation feature, any leading tabs will be replaced by equivalent spaces except, of course, those found in verbatimEnvironments (see Listing 38 on page 29) or noIndentBlock (see Listing 44 on page 30).
### Table 2: Code blocks known to `latexindent.pl`

<table>
<thead>
<tr>
<th>Code block</th>
<th>characters allowed in name</th>
<th>example</th>
</tr>
</thead>
<tbody>
<tr>
<td>environments</td>
<td>a-zA-Z*0-9___</td>
<td><code>\begin{myenv}</code> body of myenv <code>\end{myenv}</code></td>
</tr>
<tr>
<td>optionalArguments</td>
<td>inherits name from parent (e.g environment name)</td>
<td>[ opt arg text ]</td>
</tr>
<tr>
<td>mandatoryArguments</td>
<td>inherits name from parent (e.g environment name)</td>
<td>{ mand arg text }</td>
</tr>
<tr>
<td>commands</td>
<td>+a-zA-Z*0-9__:</td>
<td><code>\mycommand{arguments}</code></td>
</tr>
<tr>
<td>keyEqualsValuesBracesBrackets</td>
<td>a-zA-Z*0-9_/\h{}:#-</td>
<td><code>my key/.style=(arguments)</code></td>
</tr>
<tr>
<td>namedGroupingBracesBrackets</td>
<td>0-9.a-zA-Z**&lt;</td>
<td><code>in(arguments)</code></td>
</tr>
<tr>
<td>UnNamedGroupingBracesBrackets</td>
<td>No name!</td>
<td>{ or [ or , or &amp; or ]</td>
</tr>
<tr>
<td>ifElseFi</td>
<td>@a-zA-Z but must begin with either <code>\if</code> of @if</td>
<td><code>\ifnum...</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>\else</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>\fi</code></td>
</tr>
<tr>
<td>items</td>
<td>User specified, see Listings 134 and 137 on page 47 and on page 48</td>
<td><code>\begin{enumerate}</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>\item ...</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>\end{enumerate}</code></td>
</tr>
<tr>
<td>specialBeginEnd</td>
<td>User specified, see Listing 138 on page 48</td>
<td><code>[</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>\ ...</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>\]</code></td>
</tr>
<tr>
<td>afterHeading</td>
<td>User specified, see Listing 165 on page 54</td>
<td><code>\chapter{title}</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>...</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>\section{title}</code></td>
</tr>
<tr>
<td>filecontents</td>
<td>User specified, see Listing 54 on page 32</td>
<td><code>\begin{filecontents}</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>...</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>\end{filecontents}</code></td>
</tr>
</tbody>
</table>
2. \texttt{indentRules} for the name of the current (\texttt{thing});

3. \texttt{noAdditionalIndent} for the type of the current (\texttt{thing});

4. \texttt{indentRulesGlobal} for the type of the current (\texttt{thing}).

Using the above list, the first piece of information to be found will be used; failing that, the value of \texttt{defaultIndent} is used. If information is found in multiple fields, the first one according to the list above will be used; for example, if information is present in both \texttt{indentRules} and \texttt{noAdditionalIndentGlobal}, then the information from \texttt{indentRules} takes priority.

We now present details for the different type of code blocks known to \texttt{latexindent.pl}, as detailed in Table 2 on the preceding page; for reference, there follows a list of the code blocks covered.

5.8.1 Environments and their arguments

5.8.2 Environments with items

5.8.3 Commands with arguments

5.8.4 ifelsefi code blocks

5.8.5 specialBeginEnd code blocks

5.8.6 afterHeading code blocks

5.8.7 The remaining code blocks

5.8.7.1 keyEqualsValuesBracesBrackets

5.8.7.2 namedGroupingBracesBrackets

5.8.7.3 UnNamedGroupingBracesBrackets

5.8.7.4 filecontents

5.8.8 Summary

5.8.1 Environments and their arguments

There are a few different YAML switches governing the indentation of environments; let’s start with the code shown in Listing 174.

\begin{outer}
\begin{myenv}
body of environment
body of environment
\end{myenv}
\end{outer}

\texttt{noAdditionalIndent: } (fields)

\textbf{example 43}

If we do not wish \texttt{myenv} to receive any additional indentation, we have a few choices available to us, as demonstrated in Listings 175 and 176.

\textbf{Listing 175: myenv-noAdd1.yaml}

\texttt{noAdditionalIndent: myenv: 1}

\textbf{Listing 176: myenv-noAdd2.yaml}

\texttt{noAdditionalIndent: myenv: body: 1}

On applying either of the following commands,
we obtain the output given in Listing 177; note in particular that the environment *myenv* has not received any *additional* indentation, but that the outer environment *has* still received indentation.

**Listing 177: myenv.tex output (using either Listing 175 or Listing 176)**

```latex
\begin{outer}
  \begin{myenv}
    body of environment
    body of environment
    body of environment
  \end{myenv}
\end{outer}
```

**example 44**

Upon changing the YAML files to those shown in Listings 178 and 179, and running either

```bash
cmh:~$ latexindent.pl myenv.tex -l myenv-noAdd3.yaml
```
```bash
cmh:~$ latexindent.pl myenv.tex -l myenv-noAdd4.yaml
```

we obtain the output given in Listing 180.

**Listing 178: myenv-noAdd3.yaml**

```yaml
noAdditionalIndent:
  myenv: 0
```

**Listing 179: myenv-noAdd4.yaml**

```yaml
noAdditionalIndent:
  myenv:
    body: 0
```

**Listing 180: myenv.tex output (using either Listing 178 or Listing 179)**

```latex
\begin{outer}
  \begin{myenv}
    body of environment
    body of environment
    body of environment
  \end{myenv}
\end{outer}
```

**example 45**

Let's now allow *myenv* to have some optional and mandatory arguments, as in Listing 181.

**Listing 181: myenv-args.tex**

```latex
\begin{outer}
  \begin{myenv}[%
    optional argument text
    optional argument text]%
    { mandatory argument text
      mandatory argument text}
    body of environment
    body of environment
    body of environment
  \end{myenv}
\end{outer}
```
Upon running

```bash
cmh:~$ latexindent.pl -l=myenv-noAdd1.yaml myenv-args.tex
```

we obtain the output shown in Listing 182; note that the optional argument, mandatory argument and body all have received no additional indent. This is because, when noAdditionalIndent is specified in 'scalar' form (as in Listing 175), then all parts of the environment (body, optional and mandatory arguments) are assumed to want no additional indent.

![Listing 182: myenv-args.tex using Listing 175](image)

example 46

We may customise noAdditionalIndent for optional and mandatory arguments of the myenv environment, as shown in, for example, Listings 183 and 184.

<table>
<thead>
<tr>
<th>Listing 183: myenv-noAdd5.yaml</th>
<th>Listing 184: myenv-noAdd6.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>noAdditionalIndent:</td>
<td>noAdditionalIndent:</td>
</tr>
<tr>
<td>myenv:</td>
<td>myenv:</td>
</tr>
<tr>
<td>body: 0</td>
<td>body: 0</td>
</tr>
<tr>
<td>optionalArguments: 1</td>
<td>optionalArguments: 0</td>
</tr>
<tr>
<td>mandatoryArguments: 0</td>
<td>mandatoryArguments: 1</td>
</tr>
</tbody>
</table>

Upon running

```bash
cmh:~$ latexindent.pl myenv.tex -l=myenv-noAdd5.yaml
cmh:~$ latexindent.pl myenv.tex -l=myenv-noAdd6.yaml
```

we obtain the respective outputs given in Listings 185 and 186. Note that in Listing 185 the text for the optional argument has not received any additional indentation, and that in Listing 186 the mandatory argument has not received any additional indentation; in both cases, the body has not received any additional indentation.

<table>
<thead>
<tr>
<th>Listing 185: myenv-args.tex using Listing 183</th>
<th>Listing 186: myenv-args.tex using Listing 184</th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{outer} \begin{myenv} [</td>
<td>\begin{outer} \begin{myenv} [</td>
</tr>
<tr>
<td>optional argument text</td>
<td>optional argument text</td>
</tr>
<tr>
<td>optional argument text] %</td>
<td>optional argument text] %</td>
</tr>
<tr>
<td>{ mandatory argument text</td>
<td>{ mandatory argument text</td>
</tr>
<tr>
<td>mandatory argument text}</td>
<td>mandatory argument text}</td>
</tr>
<tr>
<td>body of environment</td>
<td>body of environment</td>
</tr>
<tr>
<td>body of environment</td>
<td>body of environment</td>
</tr>
<tr>
<td>body of environment</td>
<td>body of environment</td>
</tr>
<tr>
<td>\end{myenv}</td>
<td>\end{myenv}</td>
</tr>
<tr>
<td>\end{outer}</td>
<td>\end{outer}</td>
</tr>
</tbody>
</table>
indentRules: (fields)

example 47

We may also specify indentation rules for environment code blocks using the `indentRules` field; see, for example, Listings 187 and 188.

<table>
<thead>
<tr>
<th>Listing 187: myenv-rules1.yaml</th>
<th>Listing 188: myenv-rules2.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>indentRules:</td>
<td>indentRules:</td>
</tr>
<tr>
<td>myenv: &quot; &quot;</td>
<td>myenv:</td>
</tr>
<tr>
<td></td>
<td>body: &quot; &quot;</td>
</tr>
</tbody>
</table>

On applying either of the following commands,

```bash
cmh:~$ latexindent.pl myenv.tex -l myenv-rules1.yaml
cmh:~$ latexindent.pl myenv.tex -l myenv-rules2.yaml
```

we obtain the output given in Listing 189; note in particular that the environment `myenv` has received one tab (from the outer environment) plus three spaces from Listing 187 or 188.

<table>
<thead>
<tr>
<th>Listing 189: myenv.tex output (using either Listing 187 or Listing 188)</th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{outer}</td>
</tr>
<tr>
<td>\begin{myenv}</td>
</tr>
<tr>
<td>\hspace{\tabwidth}body_of_environment</td>
</tr>
<tr>
<td>\hspace{\tabwidth}body_of_environment</td>
</tr>
<tr>
<td>\hspace{\tabwidth}body_of_environment</td>
</tr>
<tr>
<td>\hspace{\tabwidth}\end{myenv}</td>
</tr>
<tr>
<td>\end{outer}</td>
</tr>
</tbody>
</table>

If you specify a field in `indentRules` using anything other than horizontal space, it will be ignored.

example 48

Returning to the example in Listing 181 that contains optional and mandatory arguments. Upon using Listing 187 as in

```bash
cmh:~$ latexindent.pl myenv-args.tex -l=myenv-rules1.yaml
```

we obtain the output in Listing 190; note that the body, optional argument and mandatory argument of `myenv` have all received the same customised indentation.

<table>
<thead>
<tr>
<th>Listing 190: myenv-args.tex using Listing 187</th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{outer}</td>
</tr>
<tr>
<td>\begin{myenv}[</td>
</tr>
<tr>
<td>\hspace{\tabwidth}optional_argument_text</td>
</tr>
<tr>
<td>\hspace{\tabwidth}optional_argument_text]</td>
</tr>
<tr>
<td>\hspace{\tabwidth}%</td>
</tr>
<tr>
<td>\hspace{\tabwidth}mandatory_argument_text</td>
</tr>
<tr>
<td>\hspace{\tabwidth}mandatory_argument_text}</td>
</tr>
<tr>
<td>\hspace{\tabwidth}body_of_environment</td>
</tr>
<tr>
<td>\hspace{\tabwidth}body_of_environment</td>
</tr>
<tr>
<td>\hspace{\tabwidth}body_of_environment</td>
</tr>
<tr>
<td>\hspace{\tabwidth}\end{myenv}</td>
</tr>
<tr>
<td>\end{outer}</td>
</tr>
</tbody>
</table>

example 49

You can specify different indentation rules for the different features using, for example, Listings 191 and 192
After running

cmh:∼$ latexindent.pl myenv-args.tex -l myenv-rules3.yaml
cmh:∼$ latexindent.pl myenv-args.tex -l myenv-rules4.yaml

then we obtain the respective outputs given in Listings 193 and 194.

Listing 193: myenv-args.tex using Listing 191

\begin{outer}
  \begin{myenv}[
    \begin{myenv}[
      \begin{myenv}[
        \begin{myenv}[
          \begin{myenv}[
            \begin{myenv}[
              \begin{myenv}
              \end{myenv}
            \end{myenv}
          \end{myenv}
        \end{myenv}
      \end{myenv}
    \end{myenv}
  \end{myenv}
\end{outer}

Listing 194: myenv-args.tex using Listing 192

\begin{outer}
  \begin{myenv}[
    \begin{myenv}[
      \begin{myenv}[
        \begin{myenv}[
          \begin{myenv}[
            \begin{myenv}
          \end{myenv}
        \end{myenv}
      \end{myenv}
    \end{myenv}
\end{outer}

Note that in Listing 193, the optional argument has only received a single space of indentation, while the mandatory argument has received the default (tab) indentation; the environment body has received three spaces of indentation.

In Listing 194, the optional argument has received the default (tab) indentation, the mandatory argument has received two tabs of indentation, and the body has received three spaces of indentation.

noAdditionalIndentGlobal: {fields}

Assuming that your environment name is not found within neither noAdditionalIndent nor indentRules, the next place that latexindent.pl will look is noAdditionalIndentGlobal, and in particular for the environments key (see Listing 195).

Listing 195: noAdditionalIndentGlobal

| 342 | noAdditionalIndentGlobal: |
| 343 | environments: 0 # 0/1 |

element 50

Let's say that you change the value of environments to 1 in Listing 195, and that you run

cmh:∼$ latexindent.pl myenv-args.tex -l env-noAdditionalGlobal.yaml
cmh:∼$ latexindent.pl myenv-args.tex -l myenv-rules1.yaml,env-noAdditionalGlobal.yaml

The respective output from these two commands are in Listings 196 and 197; in Listing 196 notice that both environments receive no additional indentation but that the arguments of myenv still do receive indentation. In Listing 197 notice that the outer environment does not receive additional indentation, but because of the settings from myenv-rules1.yaml (in Listing 187 on
page 61), the myenv environment still does receive indentation.

```latex
\begin{outer}
\begin{myenv}[%
    optional argument text
    optional argument text\%
    \{ mandatory argument text
    \} mandatory argument text\%
    body of environment
    body of environment
    body of environment
\end{myenv}
\end{outer}
```

**example 51**

In fact, noAdditionalIndentGlobal also contains keys that control the indentation of optional and mandatory arguments; on referencing Listings 198 and 199

```yaml
noAdditionalIndentGlobal:
    optionalArguments: 1
```

```yaml
noAdditionalIndentGlobal:
    mandatoryArguments: 1
```

we may run the commands

```
cmh:~$ latexindent.pl myenv-args.tex -local opt-args-no-add-glob.yaml
cmh:~$ latexindent.pl myenv-args.tex -local mand-args-no-add-glob.yaml
```

which produces the respective outputs given in Listings 200 and 201. Notice that in Listing 200 the optional argument has not received any additional indentation, and in Listing 201 the mandatory argument has not received any additional indentation.

```latex
\begin{outer}
\begin{myenv}[%
    optional argument text
    optional argument text\%
    \{ mandatory argument text
    \} mandatory argument text\%
    body of environment
    body of environment
    body of environment
\end{myenv}
\end{outer}
```

```yaml
indentRulesGlobal: \{fields\}
```

The final check that latexindent.pl will make is to look for indentRulesGlobal as detailed in Listing 202.

```yaml
  358 indentRulesGlobal:
  359    environments: 0  # 0/h-space
```
example 52

If you change the environments field to anything involving horizontal space, say " ", and then run the following commands

```
cmh:~$ latexindent.pl myenv-args.tex -l env-indentRules.yaml
cmh:~$ latexindent.pl myenv-args.tex -l myenv-rules1.yaml,env-indentRules.yaml
```

then the respective output is shown in Listings 203 and 204. Note that in Listing 203, both the environment blocks have received a single-space indentation, whereas in Listing 204 the outer environment has received single-space indentation (specified by indentRulesGlobal), but myenv has received " ", as specified by the particular indentRules for myenv Listing 187 on page 61.

Listing 203: myenv-args.tex using Listing 202

```
\begin{outer}
  \begin{myenv}\%
  \hspace{\textwidth}optional\hspace{-\textwidth}arg\hspace{\textwidth}text
  \hspace{\textwidth}optional\hspace{-\textwidth}arg\hspace{\textwidth}text\%
  \{\hspace{\textwidth}mand\hspace{-\textwidth}arg\hspace{\textwidth}text\%
  \hspace{\textwidth}mand\hspace{-\textwidth}arg\hspace{\textwidth}text\%
  \}
  body of environment
  \hspace{\textwidth}body of environment
  \hspace{\textwidth}body of environment
  \end{myenv}
\end{outer}
```

Listing 204: myenv-args.tex using Listings 187 and 202

```
\begin{outer}
  \begin{myenv}\%
  \hspace{\textwidth}optional\hspace{-\textwidth}arg\hspace{\textwidth}text
  \hspace{\textwidth}optional\hspace{-\textwidth}arg\hspace{\textwidth}text\%
  \{\hspace{\textwidth}mand\hspace{-\textwidth}arg\hspace{\textwidth}text\%
  \hspace{\textwidth}mand\hspace{-\textwidth}arg\hspace{\textwidth}text\%
  \}
  body of environment
  \hspace{\textwidth}body of environment
  \hspace{\textwidth}body of environment
  \end{myenv}
\end{outer}
```

example 53

You can specify indentRulesGlobal for both optional and mandatory arguments, as detailed in Listings 205 and 206

```
Listing 205: opt-args-indent-rules-glob.yaml
indentRulesGlobal:
  optionalArguments: "\t\t"
```

```
Listing 206: mand-args-indent-rules-glob.yaml
indentRulesGlobal:
  mandatoryArguments: "\t\t"
```

Upon running the following commands

```
cmh:~$ latexindent.pl myenv-args.tex -local opt-args-indent-rules-glob.yaml
cmh:~$ latexindent.pl myenv-args.tex -local mand-args-indent-rules-glob.yaml
```

we obtain the respective outputs in Listings 207 and 208. Note that the optional argument in Listing 207 has received two tabs worth of indentation, while the mandatory argument has done so in Listing 208.
5.8 noAdditionalIndent and indentRules

5.8.2 Environments with items

With reference to Listings 134 and 137 on page 47 and on page 48, some commands may contain item commands; for the purposes of this discussion, we will use the code from Listing 135 on page 47.

Assuming that you’ve populated itemNames with the name of your item, you can put the item name into noAdditionalIndent as in Listing 209, although a more efficient approach may be to change the relevant field in itemNames to 0.

example 54

Similarly, you can customise the indentation that your item receives using indentRules, as in Listing 210

Upon running the following commands

```
$ latexindent .pl items1.tex -local item-noAdd1.yaml
$ latexindent .pl items1.tex -local item-rules1.yaml
```

the respective outputs are given in Listings 211 and 212; note that in Listing 211 that the text after each item has not received any additional indentation, and in Listing 212, the text after each item has received a single space of indentation, specified by Listing 210.

```
\begin{itemize}
\item some text here
\item some more text here
\item another item
\end{itemize}
```

example 55

Alternatively, you might like to populate noAdditionalIndentGlobal or indentRulesGlobal using the items key, as demonstrated in Listings 213 and 214. Note that there is a need to ‘reset/remove’ the item field from indentRules in both cases (see the hierarchy description
given on page 56) as the item command is a member of indentRules by default.

Upon running the following commands,

```
cmh:$ latexindent.pl items1.tex -l items-noAdditionalGlobal.yaml
```
```
cmh:$ latexindent.pl items1.tex -l items-indentRulesGlobal.yaml
```

the respective outputs from Listings 211 and 212 are obtained; note, however, that all such item commands without their own individual noAdditionalIndent or indentRules settings would behave as in these listings.

### 5.8.3 Commands with arguments

**example 56**

Let's begin with the simple example in Listing 215; when latexindent.pl operates on this file, the default output is shown in Listing 216.

```
\mycommand
{ 
 mand arg text
 mand arg text}
[ 
 opt arg text
 opt arg text
]
```

As in the environment-based case (see Listings 175 and 176 on page 58) we may specify noAdditionalIndent either in 'scalar' form, or in 'field' form, as shown in Listings 217 and 218

```
\mycommand
{ 
 mand arg text
 mand arg text}
[ 
 opt arg text
 opt arg text
]
```

After running the following commands,

```
cmh:$ latexindent.pl mycommand.tex -l mycommand-noAdd1.yaml
```
```
cmh:$ latexindent.pl mycommand.tex -l mycommand-noAdd2.yaml
```

we receive the respective output given in Listings 219 and 220.
Note that in Listing 219 that the ‘body’, optional argument and mandatory argument have all received no additional indentation, while in Listing 220, only the ‘body’ has not received any additional indentation. We define the ‘body’ of a command as any lines following the command name that include its optional or mandatory arguments.

\[\text{Listing 219: mycommand.tex using Listing 217}\]
\[
\text{\textbackslash mycommand}\{\text{\textbackslash mand arg text}\text{\textbackslash mand arg text}}\}\[\text{\textbackslash opt arg text}\text{\textbackslash opt arg text}\\%]
\]
\[\text{Listing 220: mycommand.tex using Listing 218}\]
\[
\text{\textbackslash mycommand}\{\text{\textbackslash mand arg text}\text{\textbackslash mand arg text}}\]\[
\text{\textbackslash opt arg text}\text{\textbackslash opt arg text}\\%]
\]

\[\text{Listing 221: mycommand-noAdd3.yaml}\]
\[
\text{\texttt{noAdditionalIndent:}}\text{\texttt{mycommand:}}\text{\texttt{body: 0}}\text{\texttt{optionalArguments: 1}}\text{\texttt{mandatoryArguments: 0}}
\]
\[\text{Listing 222: mycommand-noAdd4.yaml}\]
\[
\text{\texttt{noAdditionalIndent:}}\text{\texttt{mycommand:}}\text{\texttt{body: 0}}\text{\texttt{optionalArguments: 0}}\text{\texttt{mandatoryArguments: 1}}
\]

After running the following commands,
\[\text{\texttt{cmh:\textbackslash$ latexindent.pl mycommand.tex -l mycommand-noAdd3.yaml}}}\]
\[\text{\texttt{cmh:\textbackslash$ latexindent.pl mycommand.tex -l mycommand-noAdd4.yaml}}}\]
we receive the respective output given in Listings 223 and 224.

\[\text{Listing 223: mycommand.tex using Listing 221}\]
\[
\text{\textbackslash mycommand}\{\text{\textbackslash mand arg text}\text{\textbackslash mand arg text}}\}\[\text{\textbackslash opt arg text}\text{\textbackslash opt arg text}\\%]
\]
\[\text{Listing 224: mycommand.tex using Listing 222}\]
\[
\text{\textbackslash mycommand}\{\text{\textbackslash mand arg text}\text{\textbackslash mand arg text}}\]\[
\text{\textbackslash opt arg text}\text{\textbackslash opt arg text}\\%]
\]

\[\text{example 57}\]
We may further customise \texttt{noAdditionalIndent} for \texttt{mycommand} as we did in Listings 183 and 184 on page 60; explicit examples are given in Listings 221 and 222.

\[\text{Listing 225: mycommand-noAdd5.yaml}\]
\[
\text{\texttt{noAdditionalIndent:}}\text{\texttt{mycommand:}}\text{\texttt{body: 0}}\text{\texttt{optionalArguments: 0}}\text{\texttt{mandatoryArguments: 1}}
\]

\[\text{example 58}\]
Attentive readers will note that the body of \texttt{mycommand} in both Listings 223 and 224 has received no additional indent, even though body is explicitly set to 0 in both Listings 221 and 222. This is because, by default, \texttt{noAdditionalIndentGlobal} for commands is set to 1 by default; this can be easily fixed as in Listings 225 and 226.
5.8 noAdditionalIndent and indentRules

After running the following commands,

```bash
cmh:~$ latexindent.pl mycommand.tex -l mycommand-noAdd5.yaml
cmh:~$ latexindent.pl mycommand.tex -l mycommand-noAdd6.yaml
```

we receive the respective output given in Listings 227 and 228.

Both indentRules and indentRulesGlobal can be adjusted as they were for environment code blocks, as in Listings 191 and 192 on page 62 and Listings 202, 205 and 206 on pages 63–64.

5.8.4 ifelsefi code blocks

**example 59**

Let's use the simple example shown in Listing 229; when latexindent.pl operates on this file, the output as in Listing 230; note that the body of each of the `\if` statements have been indented, and that the `\else` statement has been accounted for correctly.

```latex
\ifodd\radius
\ifnum\radius<14
\pgfmathparse{100-(\radius)*4};
\else
\pgfmathparse{200-(\radius)*3};
\fi\fi
```

It is recommended to specify noAdditionalIndent and indentRules in the ‘scalar’ form only for these type of code blocks, although the ‘field’ form would work, assuming that body was specified. Examples are shown in Listings 231 and 232.

```yaml
noAdditionalIndent:
  ifnum: 1
indentRules:
  ifnum: " "
```
After running the following commands,

```
cmh:~$ latexindent.pl ifelsefi1.tex -local ifnum-noAdd.yaml
cmh:~$ latexindent.pl ifelsefi1.tex -l ifnum-indent-rules.yaml
```

we receive the respective output given in Listings 233 and 234; note that in Listing 233, the `ifnum` code block has not received any additional indentation, while in Listing 234, the `ifnum` code block has received one tab and two spaces of indentation.

**Listing 233: ifelsefi1.tex using Listing 231**

```latex
\ifodd\radius
  \ifnum\radius<14
    \pgfmathparse{100-(\radius)*4};
  \else
    \pgfmathparse{200-(\radius)*3};
  \fi
\fi
```

**Listing 234: ifelsefi1.tex using Listing 232**

```latex
\ifodd\radius
  \ifnum\radius<14
    \pgfmathparse{100-(\radius)*4};
  \else
    \pgfmathparse{200-(\radius)*3};
  \fi
\fi
```

**Example 60**

We may specify `noAdditionalIndentGlobal` and `indentRulesGlobal` as in Listings 235 and 236.

**Listing 235: ifelsefi-noAdd-glob.yaml**

```
noAdditionalIndentGlobal:
  ifElseFi: 1
```

**Listing 236: ifelsefi-indent-rules-global.yaml**

```
indentRulesGlobal:
  ifElseFi: " "
```

Upon running the following commands

```
cmh:~$ latexindent.pl ifelsefi1.tex -local ifelsefi-noAdd-glob.yaml
cmh:~$ latexindent.pl ifelsefi1.tex -l ifelsefi-indent-rules-global.yaml
```

we receive the outputs in Listings 237 and 238; notice that in Listing 237 neither of the `ifelsefi` code blocks have received indentation, while in Listing 238 both code blocks have received a single space of indentation.

**Listing 237: ifelsefi1.tex using Listing 235**

```latex
\ifodd\radius
  \ifnum\radius<14
    \pgfmathparse{100-(\radius)*4};
  \else
    \pgfmathparse{200-(\radius)*3};
  \fi
\fi
```

**Listing 238: ifelsefi1.tex using Listing 236**

```latex
\ifodd\radius
  \ifnum\radius<14
    \pgfmathparse{100-(\radius)*4};
  \else
    \pgfmathparse{200-(\radius)*3};
  \fi
\fi
```

**Example 61**

We can further explore the treatment of `ifElseFi` code blocks in Listing 239, and the associated default output given in Listing 240; note, in particular, that the bodies of each of the 'or statements' have been indented.
5.8  noAdditionalIndent and indentRules

### Listing 239: ifelsefi2.tex

```
\ifcase#1
zero\%
or
one\%
or
two\%
or
three\%
\else
default
\fi
```

### Listing 240: ifelsefi2.tex default output

```
\ifcase#1
zero\%
or
one\%
or
two\%
or
three\%
\else
default
\fi
```

---

5.8.5 specialBeginEnd code blocks

Let’s use the example from Listing 139 on page 48 which has default output shown in Listing 140 on page 48.

**example 62**

It is recommended to specify noAdditionalIndent and indentRules in the 'scalar' form for these type of code blocks, although the ‘field’ form would work, assuming that body was specified. Examples are shown in Listings 241 and 242.

#### Listing 241: displayMath-noAdd.yaml

```yaml
noAdditionalIndent:
  displayMath: 1
```

#### Listing 242: displayMath-indent-rules.yaml

```yaml
indentRules:
  displayMath: \t\t\t
```

After running the following commands,

```
cmh:~$ latexindent.pl special1.tex -local displayMath-noAdd.yaml
cmh:~$ latexindent.pl special1.tex -l displayMath-indent-rules.yaml
```

we receive the respective output given in Listings 243 and 244; note that in Listing 243, the displayMath code block has not received any additional indentation, while in Listing 244, the displayMath code block has received three tabs worth of indentation.

#### Listing 243: special1.tex using Listing 241

```
The function $f$ has formula
\[
f(x)=x^2.
\]
If you like splitting dollars,
$g(x)=f(2x)$
```

#### Listing 244: special1.tex using Listing 242

```
The function $f$ has formula
\[
\begin{array}{l}
f(x)=x^2.
\end{array}
\]
If you like splitting dollars,
$g(x)=f(2x)$
```

**example 63**

We may specify noAdditionalIndentGlobal and indentRulesGlobal as in Listings 245 and 246.

#### Listing 245: special-noAdd-glob.yaml

```yaml
noAdditionalIndentGlobal:
  specialBeginEnd: 1
```

#### Listing 246: special-indent-rules-global.yaml

```yaml
indentRulesGlobal:
  specialBeginEnd: " "
```
Upon running the following commands

```
cmh:~$ latexindent.pl special1.tex -local special-noAdd-glob.yaml
cmh:~$ latexindent.pl special1.tex -l special-indent-rules-global.yaml
```

we receive the outputs in Listings 247 and 248; notice that in Listing 247 neither of the special code blocks have received indentation, while in Listing 248 both code blocks have received a single space of indentation.

<table>
<thead>
<tr>
<th>Listing 247: special1.tex using Listing 245</th>
<th>Listing 248: special1.tex using Listing 246</th>
</tr>
</thead>
</table>
| The function \$f\$ has formula \[ f(x)=x^2. \] \[ \] If you like splitting dollars, $g(x)=f(2x)$ $ | The function \$f\$ has formula \[ f(x)=x^2. \] \[
 If you like splitting dollars, $g(x)=f(2x)$ $ |

5.8.6 afterHeading code blocks

Let's use the example Listing 249 for demonstration throughout this Section. As discussed on page 55, by default `latexindent.pl` will not add indentation after headings.

```
\paragraph{paragraph title}
paragraph text
paragraph text
```

**example 64**

On using the YAML file in Listing 251 by running the command

```
cmh:~$ latexindent.pl headings2.tex -l headings3.yaml
```

we obtain the output in Listing 250. Note that the argument of `paragraph` has received (default) indentation, and that the body after the heading statement has received (default) indentation.

```
\paragraph{paragraph title}
paragraph text
paragraph text
```

<table>
<thead>
<tr>
<th>Listing 250: headings2.tex using Listing 251</th>
</tr>
</thead>
</table>
| \paragraph{paragraph title}
| paragraph text
| paragraph text

If we specify `noAdditionalIndent` as in Listing 253 and run the command

```
cmh:~$ latexindent.pl headings2.tex -l headings4.yaml
```

then we receive the output in Listing 252. Note that the arguments and the body after the heading of `paragraph` has received no additional indentation, because we have specified `noAdditionalIndent` in scalar form.
example 65

Similarly, if we specify `indentRules` as in Listing 255 and run analogous commands to those above, we receive the output in Listing 254; note that the body, mandatory argument and content after the heading of paragraph have all received three tabs worth of indentation.

```
\paragraph{paragraph title}
paragraph text
paragraph text
```

example 66

We may, instead, specify `noAdditionalIndent` in 'field' form, as in Listing 257 which gives the output in Listing 256.

```
\paragraph{paragraph title}
paragraph text
paragraph text
```

example 67

Analogously, we may specify `indentRules` as in Listing 259 which gives the output in Listing 258; note that mandatory argument text has only received a single space of indentation, while the body after the heading has received three tabs worth of indentation.

```
\paragraph{paragraph title}
paragraph text
paragraph text
```

example 68

Finally, let's consider `noAdditionalIndentGlobal` and `indentRulesGlobal` shown in Listings 261 and 263 respectively, with respective output in Listings 260 and 262. Note that in Listing 261...
the mandatory argument of paragraph has received a (default) tab's worth of indentation, while the body after the heading has received no additional indentation. Similarly, in Listing 262, the argument has received both a (default) tab plus two spaces of indentation (from the global rule specified in Listing 263), and the remaining body after paragraph has received just two spaces of indentation.

\begin{lstlisting}[language=TeX]
\paragraph{paragraph title}
paragraph text
paragraph text
\end{lstlisting}

\begin{lstlisting}[language=YAML]
indentAfterHeadings:
  paragraph:
    indentAfterThisHeading: 1
    level: 1
noAdditionalIndentGlobal:
  afterHeading: 1
\end{lstlisting}

\begin{lstlisting}[language=TeX]
\paragraph{paragraph title}
  \paragraph{paragraph text}
\end{lstlisting}

\begin{lstlisting}[language=YAML]
indentAfterHeadings:
  paragraph:
    indentAfterThisHeading: 1
    level: 1
indentRulesGlobal:
  afterHeading: " "
\end{lstlisting}

5.8.7 The remaining code blocks

Referencing the different types of code blocks in Table 2 on page 57, we have a few code blocks yet to cover; these are very similar to the \texttt{commands} code block type covered comprehensively in Section 5.8.3 on page 66, but a small discussion defining these remaining code blocks is necessary.

5.8.7.1 keyequalsValuesBracesBrackets

latexindent.pl defines this type of code block by the following criteria:

- it must immediately follow either \{ OR \[ OR \, with comments and blank lines allowed.
- then it has a name made up of the characters detailed in Table 2 on page 57;
- then an = symbol;
- then at least one set of curly braces or square brackets (comments and line breaks allowed throughout).

See the \texttt{keyEqualsValuesBracesBrackets: follow} and \texttt{keyEqualsValuesBracesBrackets: name} fields of the fine tuning section in Listing 567 on page 147.

example 69

An example is shown in Listing 264, with the default output given in Listing 265.

\begin{lstlisting}[language=TeX]
\pgfkeys{/tikz/.cd, start coordinate/.initial={0, \vertfactor},}
\end{lstlisting}

\begin{lstlisting}[language=TeX]
\pgfkeys{/tikz/.cd, __start coordinate/.initial={0, __\vertfactor},}
\end{lstlisting}

In Listing 265, note that the maximum indentation is three tabs, and these come from:

- the \texttt{\pgfkeys} command's mandatory argument;
- the start coordinate/.initial key's mandatory argument;
- the start coordinate/.initial key's body, which is defined as any lines following the name of the key that include its arguments. This is the part controlled by the \texttt{body} field for
noAdditionalIndent and friends from page 56.

5.8.7.2 namedGroupingBracesBrackets

This type of code block is mostly motivated by tikz-based code; we define this code block as follows:

- it must immediately follow either horizontal space OR one or more line breaks OR \{ OR \[ OR \$ OR \) OR (;
- the name may contain the characters detailed in Table 2 on page 57;
- then at least one set of curly braces or square brackets (comments and line breaks allowed throughout).

See the NamedGroupingBracesBrackets: follow and NamedGroupingBracesBrackets: name fields of the fine tuning section in Listing 567 on page 147.

example 70

A simple example is given in Listing 266, with default output in Listing 267.

```
\coordinate
child[grow=down]{
  edge from parent [antiparticle]
  node [above=3pt] {\$C\$
    

In particular, latexindent.pl considers child, parent and node all to be namedGroupingBracesBrackets. Referencing Listing 267, note that the maximum indentation is two tabs, and these come from:

- the child’s mandatory argument;
- the child’s body, which is defined as any lines following the name of the namedGroupingBracesBrackets that include its arguments. This is the part controlled by the body field for noAdditionalIndent and friends from page 56.

\[You may like to verify this by using the \texttt{-tt} option and checking indent.log!\]

5.8.7.3 UnNamedGroupingBracesBrackets

occur in a variety of situations; specifically, we define this type of code block as satisfying the following criteria:

- it must immediately follow either \{ OR \[ OR \$ OR \) OR (;
- then at least one set of curly braces or square brackets (comments and line breaks allowed throughout).

See the UnNamedGroupingBracesBrackets: follow field of the fine tuning section in Listing 567 on page 147.

example 71

An example is shown in Listing 268 with default output given in Listing 269.

```
\psforeach{\row}{%
{3,2.8,2.7,3,3.1},% 
{2.8,1,1.2,2,3},%
}%
```

Referencing Listing 269, there are three sets of unnamed braces. Note also that the maximum value of indentation is three tabs, and these come from:
5.9 Commands and the strings between their arguments

The command code blocks will always look for optional (square bracketed) and mandatory (curly braced) arguments which can contain comments, line breaks and ‘beamer’ commands `<.*?>` between them. There are switches that can allow them to contain other strings, which we discuss next.

| commandCodeBlocks: (fields) |

The commandCodeBlocks field contains a few switches detailed in Listing 272.
5.9 Commands and the strings between their arguments

example 72

The need for this field was mostly motivated by commands found in code used to generate images in PSTricks and tikz; for example, let’s consider the code given in Listing 273.

\begin{Verbatim}
\defFunction[algebraic]{torus}(u,v)
{(2+\cos(u))\ast \cos(v+\Pi)}
{(2+\cos(u))\ast \sin(v+\Pi)}
{\sin(u)}
\end{Verbatim}

Notice that the \defFunction command has an optional argument, followed by a mandatory argument, followed by a round-parenthesis argument, \((u,v)\).

By default, because roundParenthesesAllowed is set to 1 in Listing 272, then \latexindent.pl will allow round parenthesis between optional and mandatory arguments. In the case of the code in Listing 273, \latexindent.pl finds all the arguments of \defFunction, both before and after \((u,v)\).

The default output from running \latexindent.pl on Listing 273 actually leaves it unchanged (see Listing 274); note in particular, this is because of noAdditionalIndentGlobal as discussed on page 67.

Upon using the YAML settings in Listing 276, and running the command

\begin{Verbatim}
$ latexindent.pl pstricks1.tex -l noRoundParentheses.yaml
\end{Verbatim}

we obtain the output given in Listing 275.

\begin{Verbatim}
\defFunction[algebraic]{torus}(u,v)
{(2+\cos(u))\ast \cos(v+\Pi)}
{(2+\cos(u))\ast \sin(v+\Pi)}
{\sin(u)}
\end{Verbatim}

Notice the difference between Listing 274 and Listing 275; in particular, in Listing 275, because round parentheses are not allowed, \latexindent.pl finds that the \defFunction command finishes at the first opening round parenthesis. As such, the remaining braced, mandatory, arguments are found to be UnNamedGroupingBracesBrackets (see Table 2 on page 57) which, by
default, assume indentation for their body, and hence the tabbed indentation in Listing 275.

**example 73**

Let’s explore this using the YAML given in Listing 278 and run the command

```
cmh:~$ latexindent.pl pstricks1.tex -l defFunction.yaml
```

then the output is as in Listing 277.

Notice in Listing 277 that the *body* of the `defFunction` command i.e., the subsequent lines containing arguments after the command name, have received the single space of indentation specified by Listing 278.

**stringsAllowedBetweenArguments:** `{fields}`

**example 74**

tikz users may well specify code such as that given in Listing 279; processing this code using latexindent.pl gives the default output in Listing 280.

With reference to Listing 272 on the preceding page, we see that the strings `to`, `node`, `++` are all allowed to appear between arguments; importantly, you are encouraged to add further names to this field as necessary. This means that when latexindent.pl processes Listing 279, it consumes:

- the optional argument `[thin]`
- the round-bracketed argument `(c)` because `roundParenthesesAllowed` is 1 by default
- the string `to` (specified in `stringsAllowedBetweenArguments`)
- the optional argument `[in=110,out=-90]`
- the string `++` (specified in `stringsAllowedBetweenArguments`)
- the round-bracketed argument `(0,-0.5cm)` because `roundParenthesesAllowed` is 1 by default
- the string `node` (specified in `stringsAllowedBetweenArguments`)
- the optional argument `[below,align=left,scale=0.5]`

**example 75**

We can explore this further, for example using Listing 282 and running the command
5.9 Commands and the strings between their arguments

\begin{verbatim}
\texttt{c mh:~\$ latexindent.pl tikz-node1.tex -l draw.yaml}
\end{verbatim}

we receive the output given in Listing 281.

\begin{center}
\begin{tabular}{|l|}
\hline
\textbf{Listing 281: tikz-node1.tex using}\texttt{draw.yaml} \\
\hline
\texttt{\draw[thin]} \\
\texttt{\(\big(c\), to[in=110,out=-90]} \\
\texttt{\(\big(+0,-0.5cm\)} \\
\texttt{\(\big(node[below,align=left,scale=0.5]\big)\texttt{.}} \\
\hline
\end{tabular}
\end{center}

Notice that each line after the `\draw` command (its 'body') in Listing 281 has been given the appropriate two-spaces worth of indentation specified in Listing 282.

Let's compare this with the output from using the YAML settings in Listing 284, and running the command

\begin{verbatim}
\texttt{c mh:~\$ latexindent.pl tikz-node1.tex -l no-strings.yaml}
\end{verbatim}

given in Listing 283.

\begin{center}
\begin{tabular}{|l|}
\hline
\textbf{Listing 283: tikz-node1.tex using}\texttt{no-strings.yaml} \\
\hline
\texttt{\draw[thin]} \\
\texttt{\(\big(c\), to[in=110,out=-90]} \\
\texttt{\(\big(+0,-0.5cm\)} \\
\texttt{\(\big(node[below,align=left,scale=0.5]\big)\texttt{.}} \\
\hline
\end{tabular}
\end{center}

In this case, \texttt{latexindent.pl} sees that:

- the `\draw` command finishes after the `(c)`, as `stringsAllowedBetweenArguments` has been set to 0 so there are no strings allowed between arguments;
- it finds a namedGroupingBracesBrackets called to (see Table 2 on page 57) with argument `\{in=110,out=-90\}`
- it finds another namedGroupingBracesBrackets but this time called node with argument `\{below,align=left,scale=0.5\}`

Referencing Listing 272 on page 76, we see that the first field in the `stringsAllowedBetweenArguments` is amalgamate and is set to 1 by default. This is for users who wish to specify their settings in multiple YAML files. For example, by using the settings in either Listing 285 or Listing 286 is equivalent to using the settings in Listing 287.
We specify `amalgamate` to be set to 0 and in which case any settings loaded prior to those specified, including the default, will be overwritten. For example, using the settings in Listing 288 means that only the strings specified in that field will be used.

```
Listing 288: amalgamate-demo3.yaml
commandCodeBlocks:
  stringsAllowedBetweenArguments:
    - amalgamate: 0
    - 'further'
    - 'settings'
```

It is important to note that the `amalgamate` field, if used, must be in the first field, and specified using the syntax given in Listings 286 to 288.

**Example 76**

We may explore this feature further with the code in Listing 289, whose default output is given in Listing 290.

```
Listing 289: for-each.tex
\foreach \x/\y in {0/1,1/2}{
  body of foreach
}
```

```
Listing 290: for-each default output
\foreach \x/\y in {0/1,1/2}{
  body of foreach
}
```

Let's compare this with the output from using the YAML settings in Listing 292, and running the command

```
cmh:~$ latexindent.pl for-each.tex -l foreach.yaml
```

given in Listing 291.

```
Listing 291: for-each.tex using Listing 292
\foreach \x/\y in {0/1,1/2}{
  body of foreach
}
```

```
Listing 292: foreach.yaml
commandCodeBlocks:
  stringsAllowedBetweenArguments:
    - amalgamate: 0
    - '\x\y'
    - 'in'
```

You might like to compare the output given in Listing 290 and Listing 291. Note, in particular, in Listing 290 that the `foreach` command has not included any of the subsequent strings, and that the braces have been treated as a `namedGroupingBracesBrackets`. In Listing 291 the `foreach` command has been allowed to have `\x/\y` and in between arguments because of the settings given in Listing 292.

**Example 77**

There are some special command names that do not fit within the names recognised by `latexindent.pl`, the first one of which is `\ifnextchar`. From the perspective of `latexindent.pl`, the whole of the text `\ifnextchar` is a command, because it is immediately followed by sets of mandatory arguments. However, without the `commandNameSpecial` field, `latexindent.pl` would not be able to label it as such, because the `[ is, necessarily, not matched by a closing ]

**Example 77**

For example, consider the sample file in Listing 293, which has default output in Listing 294.
Notice that in Listing 294 the \parbox command has been able to indent its body, because \latexindent.pl has successfully found the command \ifnextchar first; the pattern-matching of \latexindent.pl starts from the inner most \texttt{<thing>} and works outwards, discussed in more detail on page 130.

For demonstration, we can compare this output with that given in Listing 295 in which the settings from Listing 296 have dictated that no special command names, including the \ifnextchar command, should not be searched for specially; as such, the \parbox command has been unable to indent its body successfully, because the \ifnextchar command has not been found.

The amalgamate field can be used for commandNameSpecial, just as for stringsAllowedBetweenArguments. The same condition holds as stated previously, which we state again here:

\textbf{Warning!}

It is important to note that the amalgamate field, if used, in either commandNameSpecial or stringsAllowedBetweenArguments must be in the first field, and specified using the syntax given in Listings 286 to 288.
SECTION 6

The -m (modifylinebreaks) switch

All features described in this section will only be relevant if the -m switch is used.

6.1 Text Wrapping ................................................. 83
  6.1.1 Text wrap: overview .................................. 83
  6.1.2 Text wrap: simple examples ...................... 84
  6.1.3 Text wrap: blocksFollow examples ............ 85
  6.1.4 Text wrap: blocksBeginWith examples ........ 89
  6.1.5 Text wrap: blocksEndBefore examples ....... 91
  6.1.6 Text wrap: trailing comments and spaces ....... 92
  6.1.7 Text wrap: when before/after .................. 94
  6.1.8 Text wrap: wrapping comments .................. 95
  6.1.9 Text wrap: huge, tabstop and separator ....... 97

6.2 oneSentencePerLine: modifying line breaks for sentences .... 98
  6.2.1 oneSentencePerLine: overview ..................... 98
  6.2.2 oneSentencePerLine: sentencesFollow .......... 101
  6.2.3 oneSentencePerLine: sentencesBeginWith .... 102
  6.2.4 oneSentencePerLine: sentencesEndWith ...... 103
  6.2.5 oneSentencePerLine: sentencesDoNOTcontain .. 104
  6.2.6 Features of the oneSentencePerLine routine .... 106
  6.2.7 oneSentencePerLine: text wrapping and indenting sentences 107
  6.2.8 oneSentencePerLine: text wrapping and indenting sentences, when before/after 110
  6.2.9 oneSentencePerLine: text wrapping sentences and comments ................. 111

6.3 Poly-switches ................................................ 111
  6.3.1 Poly-switches for environments ................. 112
    6.3.1.1 Adding line breaks: BeginStartsOnOwnLine and BodyStartsOnOwn-Line .............. 112
    6.3.1.2 Adding line breaks: EndStartsOnOwnLine and EndFinishesWithLine-Break .............. 114
    6.3.1.3 poly-switches 1, 2, and 3 only add line breaks when necessary ... 117
    6.3.1.4 Removing line breaks (poly-switches set to -1) .......... 117
    6.3.1.5 About trailing horizontal space .................. 119
    6.3.1.6 poly-switch line break removal and blank lines ............ 119
  6.3.2 Poly-switches for double backslash .............. 121
    6.3.2.1 Double backslash starts on own line ........ 121
As of Version 3.0, \texttt{latexindent.pl} has the \texttt{-m} switch, which permits \texttt{latexindent.pl} to modify line breaks, according to the specifications in the \texttt{modifyLineBreaks} field. \textit{The settings in this field will only be considered if the \texttt{-m} switch has been used}. A snippet of the default settings of this field is shown in Listing 297.

\begin{Verbatim}
modifyLineBreaks:
\end{Verbatim}

\begin{Verbatim}
\begin{verbatim}
modifyLineBreaks:
preserveBlankLines: 1 # 0/1
condenseMultipleBlankLinesInto: 1 # 0/1
\end{verbatim}
\end{Verbatim}

Listing 297: \texttt{modifyLineBreaks}

Having read the previous paragraph, it should sound reasonable that, if you call \texttt{latexindent.pl} using the \texttt{-m} switch, then you give it permission to modify line breaks in your file, but let's be clear:

\begin{Verbatim}
\begin{center}
\begin{tcolorbox}
\textbf{Warning!}

If you call \texttt{latexindent.pl} with the \texttt{-m} switch, then you are giving it permission to modify line breaks. By default, the only thing that will happen is that multiple blank lines will be condensed into one blank line; many other settings are possible, discussed next.
\end{tcolorbox}
\end{center}
\end{Verbatim}

\begin{Verbatim}
preserveBlankLines: 0|1
\end{Verbatim}

This field is directly related to \textit{poly-switches}, discussed in Section 6.3. By default, it is set to 1, which means that blank lines will be \textit{protected} from removal; however, regardless of this setting, multiple blank lines can be condensed if \texttt{condenseMultipleBlankLinesInto} is greater than 0, discussed next.

\begin{Verbatim}
condenseMultipleBlankLinesInto: (positive integer)
\end{Verbatim}

Assuming that this switch takes an integer value greater than 0, \texttt{latexindent.pl} will condense multiple blank lines into the number of blank lines illustrated by this switch.

\begin{Verbatim}
example 78
\end{Verbatim}

As an example, Listing 298 shows a sample file with blank lines; upon running

\begin{Verbatim}
$ latexindent.pl myfile.tex -m -o=+-mod1
\end{Verbatim}

the output is shown in Listing 299; note that the multiple blank lines have been condensed into one blank line, and note also that we have used the \texttt{-m} switch!
6.1 Text Wrapping

The text wrapping routine has been over-hauled as of V3.16; I hope that the interface is simpler, and most importantly, the results are better.

The complete settings for this feature are given in Listing 300.

<table>
<thead>
<tr>
<th>LISTING 298: mlb1.tex</th>
<th>LISTING 299: mlb1-mod1.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>before blank line</td>
<td>before blank line</td>
</tr>
<tr>
<td>after blank line</td>
<td>after blank line</td>
</tr>
<tr>
<td>after blank line</td>
<td>after blank line</td>
</tr>
</tbody>
</table>

**LISTING 300: textWrapOptions**

<table>
<thead>
<tr>
<th>Line</th>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>531</td>
<td>textWrapOptions:</td>
<td></td>
</tr>
<tr>
<td>532</td>
<td>columns: 0</td>
<td></td>
</tr>
<tr>
<td>533</td>
<td>multipleSpacesToSingle: 1</td>
<td></td>
</tr>
<tr>
<td>534</td>
<td>removeBlockLineBreaks: 1</td>
<td></td>
</tr>
<tr>
<td>535</td>
<td>when: before</td>
<td># before/after</td>
</tr>
<tr>
<td>536</td>
<td>comments:</td>
<td></td>
</tr>
<tr>
<td>537</td>
<td>wrap: 0</td>
<td># 0/1</td>
</tr>
<tr>
<td>538</td>
<td>inheritLeadingSpace: 0</td>
<td># 0/1</td>
</tr>
<tr>
<td>539</td>
<td>blocksFollow:</td>
<td></td>
</tr>
<tr>
<td>540</td>
<td>headings: 1</td>
<td># 0/1</td>
</tr>
<tr>
<td>541</td>
<td>commentOnPreviousLine: 1</td>
<td># 0/1</td>
</tr>
<tr>
<td>542</td>
<td>par: 1</td>
<td># 0/1</td>
</tr>
<tr>
<td>543</td>
<td>blankLine: 1</td>
<td># 0/1</td>
</tr>
<tr>
<td>544</td>
<td>verbatim: 1</td>
<td># 0/1</td>
</tr>
<tr>
<td>545</td>
<td>filecontents: 1</td>
<td># 0/1</td>
</tr>
<tr>
<td>546</td>
<td>other: \item(?:\h</td>
<td>[)</td>
</tr>
<tr>
<td>547</td>
<td>blocksBeginWith:</td>
<td></td>
</tr>
<tr>
<td>548</td>
<td>A-Z: 1</td>
<td># 0/1</td>
</tr>
<tr>
<td>549</td>
<td>a-z: 1</td>
<td># 0/1</td>
</tr>
<tr>
<td>550</td>
<td>0-9: 0</td>
<td># 0/1</td>
</tr>
<tr>
<td>551</td>
<td>other: 0</td>
<td># regex</td>
</tr>
<tr>
<td>552</td>
<td>blocksEndBefore:</td>
<td></td>
</tr>
<tr>
<td>553</td>
<td>commentOnOwnLine: 1</td>
<td># 0/1</td>
</tr>
<tr>
<td>554</td>
<td>verbatim: 1</td>
<td># 0/1</td>
</tr>
<tr>
<td>555</td>
<td>filecontents: 1</td>
<td># 0/1</td>
</tr>
<tr>
<td>556</td>
<td>other: \begin{}\[\end}</td>
<td># regex</td>
</tr>
<tr>
<td>557</td>
<td>huge: overflow</td>
<td># forbid mid-word line breaks</td>
</tr>
<tr>
<td>558</td>
<td>separator: &quot;&quot;</td>
<td></td>
</tr>
</tbody>
</table>

6.1.1 Text wrap: overview

An overview of how the text wrapping feature works:

1. the default value of columns is 0, which means that text wrapping will not happen by default;
2. it happens after verbatim blocks have been found;
3. it happens after the oneSentencePerLine routine (see Section 6.2);
4. it can happen before or after all of the other code blocks are found and does not operate on a per-code-block basis; when using before this means that, including indentation, you may receive a column width wider than that which you specify in columns, and in which case you probably wish to explore after in Section 6.1.7;
5. code blocks to be text wrapped will:
6.1 Text Wrapping

(a) follow the fields specified in blocks
(b) begin with the fields specified in blocks
(c) end before the fields specified in blocks

6. setting columns to a value > 0 will text wrap blocks by first removing line breaks, and then wrapping according to the specified value of columns;

7. setting columns to −1 will only remove line breaks within the text wrap block;

8. by default, the text wrapping routine will remove line breaks within text blocks because removeBlockLineBreaks is set to 1; switch it to 0 if you wish to change this;

9. about trailing comments within text wrap blocks:
   (a) trailing comments that do not have leading space instruct the text wrap routine to connect the lines without space (see Listing 338);
   (b) multiple trailing comments will be connected at the end of the text wrap block (see Listing 342);
   (c) the number of spaces between the end of the text wrap block and the (possibly combined) trailing comments is determined by the spaces (if any) at the end of the text wrap block (see Listing 344);

10. trailing comments can receive text wrapping; examples are shown in Section 6.1.8 and Section 6.2.9.

We demonstrate this feature using a series of examples.

6.1.2 Text wrap: simple examples

example 79

Let’s use the sample text given in Listing 301.

**Listing 301: textwrap1.tex**

Here is a line of text that will be wrapped by latexindent.pl.

Here is a line of text that will be wrapped by latexindent.pl.

We will change the value of columns in Listing 303 and then run the command

```bash
$ latexindent.pl -m -l textwrap1.yaml textwrap1.tex
```

then we receive the output given in Listing 302.

**Listing 302: textwrap1-mod1.tex**

Here is a line of text that will be wrapped by latexindent.pl.

Here is a line of text that will be wrapped by latexindent.pl.

**Listing 303: textwrap1.yaml**

```yaml
modifyLineBreaks:
  textWrapOptions:
    columns: 20
```

example 80

If we set columns to −1 then latexindent.pl remove line breaks within the text wrap block, and will not perform text wrapping. We can use this to undo text wrapping.
Starting from the file in Listing 302 and using the settings in Listing 304

```
LISTING 304: textwrap1A.yaml
modifyLineBreaks:
  textWrapOptions:
    columns: -1
```

and running

```
$ latexindent.pl -m -l textwrap1A.yaml textwrap1-mod1.tex
```

gives the output in Listing 305.

```
LISTING 305: textwrap1-mod1A.tex
Here is a line of text that will be wrapped by latexindent.pl.
Here is a line of text that will be wrapped by latexindent.pl.
```

**example 81**

By default, the text wrapping routine will convert multiple spaces into single spaces. You can change this behaviour by flicking the switch `multipleSpacesToSingle` which we have done in Listing 307

Using the settings in Listing 307 and running

```
$ latexindent.pl -m -l textwrap1B.yaml textwrap1-mod1.tex
```

gives the output in Listing 306.

```
LISTING 306: textwrap1-mod1B.tex
Here is a line of text that will be wrapped by latexindent.pl.
Here is a line of text that will be wrapped by latexindent.pl.
```

```
LISTING 307: textwrap1B.yaml
modifyLineBreaks:
  textWrapOptions:
    columns: 20
    multipleSpacesToSingle: 0
```

We note that in Listing 306 the multiple spaces have *not* been condensed into single spaces.

### 6.1.3 Text wrap: blocksFollow examples

We examine the `blocksFollow` field of Listing 300.

**example 82**

Let's use the sample text given in Listing 308.
We note that Listing 308 contains the heading commands section and subsection. Upon running the command

```bash
cmh:~$ latexindent.pl -m -l textwrap1.yaml tw-headings1.tex
```
then we receive the output given in Listing 309.

We reference Listing 300 on page 83 and also Listing 165 on page 54:

- in Listing 300 the headings field is set to 1, which instructs latexindent.pl to read the fields from Listing 165 on page 54, regardless of the value of indentAfterThisHeading or level;
- the default is to assume that the heading command can, optionally, be followed by a label command.

If you find scenarios in which the default value of headings does not work, then you can explore the other field.

We can turn off headings as in Listing 311 and then run

```bash
cmh:~$ latexindent.pl -m -l textwrap1.yaml,bf-no-headings.yaml tw-headings1.tex
```
gives the output in Listing 310, in which text wrapping has been instructed not to happen following headings.

Let’s use the sample text given in Listing 312.
6.1 Text Wrapping

Listing 312: tw-comments1.tex

% trailing comment
text to be wrapped following first comment
% another comment
text to be wrapped following second comment

We note that Listing 312 contains trailing comments. Upon running the command

```
cmh:~$ latexindent.pl -m -l textwrap1.yaml tw-comments1.tex
```
then we receive the output given in Listing 313.

Listing 313: tw-comments1-mod1.tex

% trailing comment
text to be wrapped following first comment
% another comment
text to be wrapped following second comment

With reference to Listing 300 on page 83 the commentOnPreviousLine field is set to 1, which instructs latexindent.pl to find text wrap blocks after a comment on its own line. We can turn off comments as in Listing 315 and then run

```
cmh:~$ latexindent.pl -m -l textwrap1.yaml,bf-no-comments.yaml tw-comments1.tex
```
gives the output in Listing 314, in which text wrapping has been instructed not to happen following comments on their own line.

Listing 314: tw-comments1-mod2.tex

% trailing comment
text to be wrapped following first comment
% another comment
text to be wrapped following second comment

Referencing Listing 300 on page 83 the blocksFollow fields par, blankline, verbatim and filecontents fields operate in analogous ways to those demonstrated in the above. The other field of the blocksFollow can either be 0 (turned off) or set as a regular expression. The default value is set to `\item(?:\h|\[)` which can be translated to `backslash followed by a square bracket or backslash item followed by horizontal space or a square bracket`, or in other words, `end of display math` or an item command.

Example 84

Let's use the sample text given in Listing 316.
We note that Listing 316 contains display math. Upon running the command

```bash
cmh:~$ latexindent.pl -m -l textwrap1.yaml tw-disp-math1.tex
```

then we receive the output given in Listing 317.

```latex
\[ y = x \]
```

With reference to Listing 300 on page 83 the other field is set to `\\`, which instructs latexindent.pl to find text wrap blocks after the end of display math.

We can turn off this switch as in Listing 319 and then run

```bash
cmh:~$ latexindent.pl -m -l textwrap1.yaml,bf-no-disp-math.yaml tw-disp-math1.tex
```

which gives the output in Listing 318, in which text wrapping has been instructed *not to happen* following display math.

```latex
\[ y = x \]
```

Naturally, you should feel encouraged to customise this as you see fit.

The `blocksFollow` field *deliberately* does not default to allowing text wrapping to occur after begin environment statements. You are encouraged to customize the other field to accommodate the environments that you would like to text wrap individually, as in the next example.

**example 85**

Let's use the sample text given in Listing 320.
6.1 Text Wrapping

We note that Listing 320 contains \texttt{myenv} environment. Upon running the command

\texttt{cmh:~$ latexindent.pl -m -l textwrap1.yaml tw-bf-myenv1.tex}

then we receive the output given in Listing 321.

We note that we have \textit{not} received much text wrapping. We can turn do better by employing Listing 323 and then run

\texttt{cmh:~$ latexindent.pl -m -l textwrap1.yaml,tw-bf-myenv.yaml tw-bf-myenv1.tex}

which gives the output in Listing 322, in which text wrapping has been implemented across the file.

6.1.4 Text wrap: \texttt{blocksBeginWith} examples

We examine the \texttt{blocksBeginWith} field of Listing 300 with a series of examples.
example 86

By default, text wrap blocks can begin with the characters a–z and A–Z.

If we start with the file given in Listing 324

```
123 text to be wrapped before display math \[ y = x \]
456 text to be wrapped after display math
```

and run the command

```
$ latexindent.pl -m -l textwrap1.yaml tw-0-9.tex
```

then we receive the output given in Listing 325 in which text wrapping has **not** occurred.

```
123 text to be wrapped before display math \[ y = x \]
456 text to be wrapped after display math
```

We can allow paragraphs to begin with 0–9 characters by using the settings in Listing 327 and running

```
$ latexindent.pl -m -l textwrap1.yaml,bb-0-9-yaml tw-0-9.tex
```

gives the output in Listing 326, in which text wrapping **has** happened.

```
123 text to be wrapped before display math \[ y = x \]
456 text to be wrapped after display math
```

example 87

Let’s now use the file given in Listing 328

```
% trailing comment
\announce{announce text}
   and text to be wrapped before goes here
```

and run the command

```
$ latexindent.pl -m 
```

Listing 327: bb-0-9.yaml.yaml
```
modifyLineBreaks:
  textWrapOptions:
    blocksBeginWith:
      0-9: 1
```
then we receive the output given in Listing 329 in which text wrapping has not occurred.

We can allow \announce to be at the beginning of paragraphs by using the settings in Listing 331 and running

```bash
cmh:~$ latexindent.pl -m -l textwrap1.yaml,tw-bb-announce.yaml tw-bb-announce1.tex
```
gives the output in Listing 330, in which text wrapping has happened.

6.1.5 Text wrap: blocksEndBefore examples

We examine the blocksEndBefore field of Listing 300 with a series of examples.

example 88

Let's use the sample text given in Listing 332.

```latex
before \begin{align} 1 & 2 \\ 3 & 4 \end{align} after \end{align}
```

We note that Listing 332 contains an environment. Upon running the command

```bash
cmh:~$ latexindent.pl -m -l textwrap1A.yaml tw-be-equation.tex
```
then we receive the output given in Listing 333.
With reference to Listing 300 on page 83 the other field is set to `\begin{align}::/\end{align}`, which instructs `latexindent.pl` to stop text wrap blocks before `begin` statements, display math, and `end` statements.

We can turn off this switch as in Listing 334 and then run

```
cmh:~$ latexindent.pl -m tw-be-equation.yaml tw-be-equation.tex
```

gives the output in Listing 335, in which text wrapping has been instructed *not* to stop at these statements.

```
LISTING 334: tw-be-equation.yaml
```

```yaml
modifyLineBreaks:
  textWrapOptions:
    blocksEndBefore:
      other: 0
```

```
LISTING 335: tw-be-equation-mod2.tex
```

```latex
\begin{align}
  1 & 2 \\
  3 & 4 \\
\end{align}
```

Naturally, you should feel encouraged to customise this as you see fit.

### 6.1.6 Text wrap: trailing comments and spaces

We explore the behaviour of the text wrap routine in relation to trailing comments using the following examples.

**example 89**

The file in Listing 336 contains a trailing comment which *does* have a space in front of it.

Running the command

```
cmh:~$ latexindent.pl -m tw-tc1.tex -l textwrap1A.yaml tw-be-equation.yam1 tw-be-equation.tex
```

gives the output given in Listing 337.

```
LISTING 336: tw-tc1.tex
```

```latex
foo␣%
bar
```

```
LISTING 337: tw-tc1-mod1.tex
```

```latex
foo bar%
```

**example 90**

The file in Listing 338 contains a trailing comment which does *not* have a space in front of it.

Running the command

```
cmh:~$ latexindent.pl -m tw-tc2.tex -l textwrap1A.yaml tw-be-equation.tex
```

6.1 Text Wrapping

gives the output in Listing 339.

<table>
<thead>
<tr>
<th>Listing 338: tw-tc2.tex</th>
<th>Listing 339: tw-tc2-mod1.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>foo%</td>
<td>foobar%</td>
</tr>
<tr>
<td>bar</td>
<td></td>
</tr>
</tbody>
</table>

We note that, because there is not a space before the trailing comment, that the lines have been joined without a space.

example 91

The file in Listing 340 contains multiple trailing comments.

Running the command

```
cmh:~$ latexindent.pl -m tw-tc3.tex -l textwrap1A.yaml -o=+-mod1
```

gives the output in Listing 341.

<table>
<thead>
<tr>
<th>Listing 340: tw-tc3.tex</th>
<th>Listing 341: tw-tc3-mod1.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>foo %1</td>
<td>foo bar three %1 %2</td>
</tr>
<tr>
<td>bar %2</td>
<td></td>
</tr>
<tr>
<td>three</td>
<td></td>
</tr>
</tbody>
</table>

example 92

The file in Listing 342 contains multiple trailing comments.

Running the command

```
cmh:~$ latexindent.pl -m tw-tc4.tex -l textwrap1A.yaml -o=+-mod1
```

gives the output in Listing 343.

<table>
<thead>
<tr>
<th>Listing 342: tw-tc4.tex</th>
<th>Listing 343: tw-tc4-mod1.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>foo %1</td>
<td>foo bar three %1 %2 %3</td>
</tr>
<tr>
<td>bar %2</td>
<td></td>
</tr>
<tr>
<td>three %3</td>
<td></td>
</tr>
</tbody>
</table>

example 93

The file in Listing 344 contains multiple trailing comments.

Running the command

```
cmh:~$ latexindent.pl -m tw-tc5.tex -l textwrap1A.yaml -o=+-mod1
```

gives the output in Listing 345.

<table>
<thead>
<tr>
<th>Listing 344: tw-tc5.tex</th>
<th>Listing 345: tw-tc5-mod1.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>foo %1</td>
<td>foobar three %1 %2 %3</td>
</tr>
<tr>
<td>bar %2</td>
<td></td>
</tr>
<tr>
<td>three %3</td>
<td></td>
</tr>
</tbody>
</table>

The space at the end of the text block has been preserved.

example 94

The file in Listing 346 contains multiple trailing comments.
Running the command

```bash
cmh:$ latexindent.pl -m tw-tc6.tex -l textwrap1A.yaml -o=+-mod1
```

gives the output in Listing 347.

<table>
<thead>
<tr>
<th>LISTING 346: tw-tc6.tex</th>
<th>LISTING 347: tw-tc6-mod1.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>foo%1 bar</td>
<td>foobar%1</td>
</tr>
</tbody>
</table>

The space at the end of the text block has been preserved.

### 6.1.7 Text wrap: when before/after

The text wrapping routine operates, by default, before the code blocks have been found, but this can be changed to after:

- **before** means it is likely that the columns of wrapped text may exceed the value specified in columns;
- **after** means it columns of wrapped text should not exceed the value specified in columns.

We demonstrate this in the following examples. See also Section 6.2.8.

**example 95**

Let's begin with the file in Listing 348.

```latex
\begin{myenv}
This paragraph has line breaks throughout its paragraph; we would like to combine the textwrapping and paragraph removal routine.
\end{myenv}
```

Using the settings given in Listing 350 and running the command

```bash
cmh:$ latexindent.pl textwrap8.tex -o=+-mod1.tex -l=tw-before1.yaml -m
```

gives the output given in Listing 349.
We note that, in Listing 349, that the wrapped text has exceeded the specified value of `columns` (35) given in Listing 350. We can affect this by changing `when`; we explore this next.

**example 96**

We continue working with Listing 348.

Using the settings given in Listing 352 and running the command

```
$ latexindent -m \texttt{textwrap8.tex} -o=+-mod2.tex -l=tw-after1.yaml -m
```

gives the output given in Listing 351.

We note that, in Listing 351, that the wrapped text has obeyed the specified value of `columns` (35) given in Listing 352.

### 6.1.8 Text wrap: wrapping comments

You can instruct `latexindent.pl` to apply text wrapping to comments; we demonstrate this with examples, see also Section 6.2.9.

**example 97**

We use the file in Listing 353 which contains a trailing comment block.
Using the settings given in Listing 355 and running the command

```
cmh:~$ latexindent.pl textwrap9.tex -o=+-mod1.tex -l=wrap-comments1.yaml -m
```

```
My first sentence
% first comment second third
% comment fourth
----|----|----|----|----|----|----|----|
 5 10 15 20 25 30 35 40
```

We note that, in Listing 354, that the comments have been *combined and wrapped* because of the annotated line specified in Listing 355.

**example 98**

We use the file in Listing 356 which contains a trailing comment block.

```
My first sentence
% first comment
% second
%third comment
% fourth
```

Using the settings given in Listing 358 and running the command

```
cmh:~$ latexindent.pl textwrap10.tex -o=+-mod1.tex -l=wrap-comments1.yaml -m
```

```
My first sentence
% first comment second third
% comment fourth
----|----|----|----|----|----|----|----|
 5 10 15 20 25 30 35 40
```

We note that, in Listing 357, that the comments have been *combined and wrapped* because of the annotated line specified in Listing 358, and that the space from the leading comment has not been inherited; we will explore this further in the next example.

**example 99**

We continue to use the file in Listing 356.

Using the settings given in Listing 360 and running the command

```
cmh:~$ latexindent.pl textwrap10.tex -o=+-mod2.tex -l=wrap-comments2.yaml -m
```

```
My first sentence
% first comment second third
% comment fourth
----|----|----|----|----|----|----|----|
 5 10 15 20 25 30 35 40
```

We note that, in Listing 359, that the comments have been *combined and wrapped* because of the annotated line specified in Listing 360, and that the space from the leading comment has not been inherited; we will explore this further in the next example.
6.1 Text Wrapping

We note that, in Listing 359, that the comments have been combined and wrapped and that the leading space has been inherited because of the annotated lines specified in Listing 360.

6.1.9 Text wrap: huge, tabstop and separator

The default value of huge is overflow, which means that words will not be broken by the text wrapping routine, implemented by the Text::Wrap [47]. There are options to change the huge option for the Text::Wrap module to either wrap or die. Before modifying the value of huge, please bear in mind the following warning:

**Warning!**

Changing the value of huge to anything other than overflow will slow down latexindent.pl significantly when the \texttt{-m} switch is active.

Furthermore, changing huge means that you may have some words or commands(!) split across lines in your .tex file, which may affect your output. I do not recommend changing this field.

example 100

For example, using the settings in Listings 362 and 364 and running the commands

```
cmh:~$ latexindent.pl -m textwrap4.tex -o=+-mod2A -l textwrap2A.yaml
cmh:~$ latexindent.pl -m textwrap4.tex -o=+-mod2B -l textwrap2B.yaml
```

gives the respective output in Listings 361 and 363.

You can also specify the tabstop field as an integer value, which is passed to the text wrap module; see [47] for details.
### 6.2 oneSentencePerLine: modifying line breaks for sentences

You can instruct latexindent.pl to format your file so that it puts one sentence per line. Thank you to [7] for helping to shape and test this feature. The behaviour of this part of the script is controlled by the switches detailed in Listing 368, all of which we discuss next.

#### 6.2.1 oneSentencePerLine: overview

An overview of how the oneSentencePerLine routine feature works:

1. the default value of manipulateSentences is 0, which means that oneSentencePerLine will not happen by default;
2. it happens after verbatim blocks have been found;
3. it happens before the text wrapping routine (see Section 6.1);
4. it happens before the main code blocks have been found;
5. sentences to be found:
   (a) follow the fields specified in sentencesFollow
   (b) begin with the fields specified in sentencesBeginWith
   (c) end with the fields specified in sentencesEndWith
6. by default, the oneSentencePerLine routine will remove line breaks within sentences because removeBlockLineBreaks is set to 1; switch it to 0 if you wish to change this;
7. sentences can be text wrapped according to textWrapSentences, and will be done either before or after the main indentation routine (see Section 6.2.8);
8. about trailing comments within text wrap blocks:
   (a) multiple trailing comments will be connected at the end of the sentence;
   (b) the number of spaces between the end of the sentence and the (possibly combined) trailing comments is determined by the spaces (if any) at the end of the sentence.

We demonstrate this feature using a series of examples.

| manipulateSentences: 0|1 |
|----------------------|
| This is a binary switch that details if latexindent.pl should perform the sentence manipulation routine; it is off (set to 0) by default, and you will need to turn it on (by setting it to 1) if you want the script to modify line breaks surrounding and within sentences. |

| removeSentenceLineBreaks: 0|1 |
|--------------------------|
| When operating upon sentences latexindent.pl will, by default, remove internal line breaks as removeSentenceLineBreaks is set to 1. Setting this switch to 0 instructs latexindent.pl not to do so. |

**example 102**

For example, consider multiple-sentences.tex shown in Listing 369.

<table>
<thead>
<tr>
<th>LISTING 369: multiple-sentences.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>This is the first sentence. This is the; second, sentence. This is the third sentence.</td>
</tr>
<tr>
<td>This is the fourth sentence! This is the fifth sentence? This is the sixth sentence.</td>
</tr>
</tbody>
</table>

If we use the YAML files in Listings 371 and 373, and run the commands

```
$ latexindent.pl multiple-sentences -m -l=manipulate-sentences.yaml
$ latexindent.pl multiple-sentences -m -l=keep-sen-line-breaks.yaml
```

then we obtain the respective output given in Listings 370 and 372.
By default, the one-sentence-per-line routine will convert multiple spaces into single spaces. You can change this behaviour by changing the switch `multipleSpacesToSingle` to a value of 0.

The remainder of the settings displayed in Listing 368 on page 98 instruct `latexindent.pl` on how to define a sentence. From the perspective of `latexindent.pl` a sentence must:

- **follow** a certain character or set of characters (see Listing 374); by default, this is either `\par`, a blank line, a full stop/period (.), exclamation mark (!), question mark (?) right brace (}) or a comment on the previous line;
- **begin** with a character type (see Listing 375); by default, this is only capital letters;
- **end** with a character (see Listing 376); by default, these are full stop/period (.), exclamation mark (!) and question mark (?).

In each case, you can specify the `other` field to include any pattern that you would like; you can specify anything in this field using the language of regular expressions.
6.2 oneSentencePerLine: modifying line breaks for sentences

### Listing 366: sentencesEndWith

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>523</td>
<td>sentencesEndWith:</td>
<td></td>
</tr>
<tr>
<td>524</td>
<td>basicFullStop:</td>
<td>0 # 0/1</td>
</tr>
<tr>
<td>525</td>
<td>betterFullStop:</td>
<td>1 # 0/1</td>
</tr>
<tr>
<td>526</td>
<td>exclamationMark:</td>
<td>1 # 0/1</td>
</tr>
<tr>
<td>527</td>
<td>questionMark:</td>
<td>1 # 0/1</td>
</tr>
<tr>
<td>528</td>
<td>other:</td>
<td>0 # regex</td>
</tr>
</tbody>
</table>

#### 6.2.2 oneSentencePerLine: sentencesFollow

Let’s explore a few of the switches in sentencesFollow.

**Example 103**

We start with Listing 369 on page 99, and use the YAML settings given in Listing 378. Using the command

```
$ latexindent.pl multiple-sentences -m -l=sentences-follow1.yaml
```

we obtain the output given in Listing 377.

#### Listing 377: multiple-sentences.tex using Listing 378

This is the first sentence.
This is the; second, sentence.
This is the third sentence.

This is the fourth sentence!
This is the fifth sentence?
This is the sixth sentence.

Notice that, because blankLine is set to 0, latexindent.pl will not seek sentences following a blank line, and so the fourth sentence has not been accounted for.

**Example 104**

We can explore the other field in Listing 374 with the .tex file detailed in Listing 379.

#### Listing 379: multiple-sentences1.tex

(Some sentences stand alone in brackets.) This is the first sentence. This is the; second, sentence. This is the third sentence.

Upon running the following commands

```
$ latexindent.pl multiple-sentences1 -m -l=manipulate-sentences.yaml
$ latexindent.pl multiple-sentences1 -m -l=manipulate-sentences.yaml,sentences-follow2.yaml
```

then we obtain the respective output given in Listings 380 and 381.

#### Listing 380: multiple-sentences1.tex using Listing 371 on the preceding page

(Some sentences stand alone in brackets.) This is the first sentence.
This is the; second, sentence.
This is the third sentence.
6.2 oneSentencePerLine: modifying line breaks for sentences

Listing 381: multiple-sentences1.tex using Listing 382

(Some sentences stand alone in brackets.)
This is the first sentence.
This is the; second, sentence.
This is the third sentence.

Notice that in Listing 380 the first sentence after the ) has not been accounted for, but that following the inclusion of Listing 382, the output given in Listing 381 demonstrates that the sentence has been accounted for correctly.

6.2.3 oneSentencePerLine: sentencesBeginWith

By default, \texttt{latexindent.pl} will only assume that sentences begin with the upper case letters A-Z; you can instruct the script to define sentences to begin with lower case letters (see Listing 375), and we can use the other field to define sentences to begin with other characters.

\textbf{example 105}

We use the file in Listing 383.

Listing 383: multiple-sentences2.tex

\begin{verbatim}
This is the first sentence.

\$a\$ can represent a number. 7 is at the beginning of this sentence.
\end{verbatim}

Upon running the following commands

\begin{verbatim}
cmh:~$ latexindent.pl multiple-sentences2 -m -l=manipulate-sentences.yaml

then we obtain the respective output given in Listings 384 and 385.

Listing 384: multiple-sentences2.tex using Listing 371 on page 100

This is the first sentence.

\$a\$ can represent a number. 7 is at the beginning of this sentence.

Listing 385: multiple-sentences2.tex using Listing 386

This is the first sentence.

\$a\$ can represent a number. 7 is at the beginning of this sentence.

Notice that in Listing 384, the first sentence has been accounted for but that the subsequent sentences have not. In Listing 385, all of the sentences have been accounted for, because the other field in Listing 386 has defined sentences to begin with either \$ or any numeric digit, 0 to 9.
6.2.4 oneSentencePerLine: sentencesEndWith

example 106

Let's return to Listing 369 on page 99; we have already seen the default way in which \texttt{latexindent.pl} will operate on the sentences in this file in Listing 370 on page 100. We can populate the other field with any character that we wish; for example, using the YAML specified in Listing 388 and the command

\begin{lstlisting}[language=sh]
\$ latexindent.pl multiple-sentences -m -l=sentences-end1.yaml
\$ latexindent.pl multiple-sentences -m -l=sentences-end2.yaml
\end{lstlisting}

then we obtain the output in Listing 387.

\begin{Verbatim}
\texttt{Listing 387: multiple-sentences.tex} using \texttt{Listing 388}
\end{Verbatim}

This is the first sentence.
This is the second, sentence.
This is the third sentence.

This is the fourth sentence!
This is the fifth sentence?
This is the sixth sentence.

\begin{Verbatim}
\texttt{Listing 389: multiple-sentences.tex} using \texttt{Listing 390}
\end{Verbatim}

This is the first sentence.
This is the second, sentence.
This is the third sentence.

This is the fourth sentence!
This is the fifth sentence?
This is the sixth sentence.

There is a subtle difference between the output in Listings 387 and 389; in particular, in Listing 387 the word sentence has not been defined as a sentence, because we have not instructed \texttt{latexindent.pl} to begin sentences with lower case letters. We have changed this by using the settings in Listing 390, and the associated output in Listing 389 reflects this.

Referencing Listing 376 on page 101, you'll notice that there is a field called basicFullStop, which is set to 0, and that the betterFullStop is set to 1 by default.

example 107

Let's consider the file shown in Listing 391.

\begin{Verbatim}
\texttt{Listing 391: url.tex}
\end{Verbatim}

This sentence, \url{tex.stackexchange.com/} finishes here. Second sentence.

Upon running the following commands

\begin{lstlisting}[language=sh]
\$ latexindent.pl url -m -l=manipulate-sentences.yaml
\end{lstlisting}

we obtain the output given in Listing 392.
6.2 oneSentencePerLine: modifying line breaks for sentences  

<table>
<thead>
<tr>
<th>Listing 392: url.tex using Listing 371 on page 100</th>
</tr>
</thead>
</table>
| This sentence, \url{tex.stackexchange.com/} finishes here.
Second sentence. |

Notice that the full stop within the url has been interpreted correctly. This is because, within the betterFullStop, full stops at the end of sentences have the following properties:

- they are ignored within e.g. and i.e.;
- they can not be immediately followed by a lower case or upper case letter;
- they can not be immediately followed by a hyphen, comma, or number.

If you find that the betterFullStop does not work for your purposes, then you can switch it off by setting it to 0, and you can experiment with the other field. You can also seek to customise the betterFullStop routine by using the fine tuning, detailed in Listing 567 on page 147.

The basicFullStop routine should probably be avoided in most situations, as it does not accommodate the specifications above.

**example 108**

For example, using the following command

```bash
$ latexindent.pl url -m -l=alt-full-stop1.yaml
```

and the YAML in Listing 394 gives the output in Listing 393.

<table>
<thead>
<tr>
<th>Listing 393: url.tex using Listing 394</th>
</tr>
</thead>
<tbody>
<tr>
<td>This sentence, \url{tex.stackexchange.com/} finishes here. Second sentence.</td>
</tr>
</tbody>
</table>

Notice that the full stop within the URL has not been accommodated correctly because of the non-default settings in Listing 394.

<table>
<thead>
<tr>
<th>Listing 394: alt-full-stop1.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>modifyLineBreaks:</td>
</tr>
<tr>
<td>oneSentencePerLine:</td>
</tr>
<tr>
<td>manipulateSentences: 1</td>
</tr>
<tr>
<td>sentencesEndWith:</td>
</tr>
<tr>
<td>basicFullStop: 1</td>
</tr>
<tr>
<td>betterFullStop: 0</td>
</tr>
</tbody>
</table>

6.2.5 oneSentencePerLine: sentencesDoNOTcontain

You can specify patterns that sentences do not contain using the field in Listing 395.

<table>
<thead>
<tr>
<th>Listing 395: sentencesDoNOTcontain</th>
</tr>
</thead>
<tbody>
<tr>
<td>529 sentencesDoNOTcontain:</td>
</tr>
<tr>
<td>530 other: \begin # regex</td>
</tr>
</tbody>
</table>

If sentences run across environments then, by default, they will not be considered a sentence by latexindent.pl.

**example 109**

For example, if we use the .tex file in Listing 396

<table>
<thead>
<tr>
<th>Listing 396: multiple-sentences4.tex</th>
</tr>
</thead>
</table>
| This sentence \begin{itemize}
\item continues
\end{itemize} across itemize and finishes here. |
and run the command

```
cmh:~$ latexindent.pl multiple-sentences4 -m -l=manipulate-sentences.yaml
```
then the output is unchanged, because the default value of sentencesDoNOTcontain says, *sentences do NOT contain*

This means that, by default, `latexindent.pl` does not consider the file in Listing 396 to have a sentence. `\begin`

**example 110**

We can customise the sentencesDoNOTcontain field with anything that we do *not* want sentences to contain.

We begin with the file in Listing 397.

**Listing 397: sentence-dnc1.tex**

```
This should not be a sentence \cmh{} and should not change. 
But this one should.
```

Upon running the following commands

```
cmh:~$ latexindent.pl sentence-dnc1.tex -m -l=dnc1.yaml
```
then we obtain the output given in Listing 398.

**Listing 398: sentence-dnc1-mod1.tex**

```
This should not be a sentence \cmh{} and should not change. 
But this one should.
```

The settings in Listing 399 say that sentences do *not* contain `\begin` and that they do not contain `\cmh`

**example 111**

We can implement case insensitivity for the sentencesDoNOTcontain field.

We begin with the file in Listing 400.

**Listing 400: sentence-dnc2.tex**

```
This should not be a sentence \cmh{} and should not change. 
This should not be a sentence \CMH{} and should not change. 
But this one should.
```

Upon running the following commands

```
cmh:~$ latexindent.pl sentence-dnc2.tex -m -l=dnc2.yaml
```
then we obtain the output given in Listing 401.
6.2 oneSentencePerLine: modifying line breaks for sentences

<table>
<thead>
<tr>
<th>LISTING 401: sentence-dnc2-mod2.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>This should not be a sentence \cmh(?) and should not change.\cmh{?} and should not change. But this one should.</td>
</tr>
</tbody>
</table>

The settings in Listing 402 say that sentences do not contain \begin and that they do not contain case insensitive versions of \cmh.

example 112

We can turn off sentencesDoNOTcontain by setting it to 0 as in Listing 403.

<table>
<thead>
<tr>
<th>LISTING 403: dnc-off.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>modifyLineBreaks:</td>
</tr>
<tr>
<td>oneSentencePerLine:</td>
</tr>
<tr>
<td>manipulateSentences: 1</td>
</tr>
<tr>
<td>sentencesDoNOTcontain: 0</td>
</tr>
</tbody>
</table>

The settings in Listing 403 mean that sentences can contain any character.

6.2.6 Features of the oneSentencePerLine routine

The sentence manipulation routine takes place after verbatim environments, preamble and trailing comments have been accounted for; this means that any characters within these types of code blocks will not be part of the sentence manipulation routine.

example 113

For example, if we begin with the .tex file in Listing 404, and run the command

```
cmh:~$ latexindent.pl multiple-sentences3 -m -l=manipulate-sentences.yaml
```

then we obtain the output in Listing 405.

<table>
<thead>
<tr>
<th>LISTING 404: multiple-sentences3.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>The first sentence continues after the verbatim \begin{verbatim}there are sentences within this. These will not be operated upon by latexindent.pl.\end{verbatim} and finishes here. Second sentence % a commented full stop. contains trailing comments, which are ignored.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LISTING 405: multiple-sentences3.tex using Listing 371 on page 100</th>
</tr>
</thead>
<tbody>
<tr>
<td>The first sentence continues after the verbatim \begin{verbatim}there are sentences within this. These will not be operated upon by latexindent.pl.\end{verbatim} and finishes here. Second sentence contains trailing comments, which are ignored. % a commented full stop.</td>
</tr>
</tbody>
</table>
6.2.7 oneSentencePerLine: text wrapping and indenting sentences

The oneSentencePerLine can be instructed to perform text wrapping and indentation upon sentences.

example 114

Let’s use the code in Listing 406.

```latex
\begin{itemize}
  \item firstly.
  \item secondly.
\end{itemize}
```

By default, \texttt{latexindent.pl} will find the full-stop within the first item, which means that, upon running the following commands

```bash
$ latexindent.pl multiple-sentences6 -m -l=sentence-wrap1.yaml
$ latexindent.pl multiple-sentences6 -m -l=sentence-wrap1.yaml
```

we receive the respective output in Listing 410 and Listing 411.
We note that Listing 410 the \texttt{itemize} code block has \textit{not} been indented appropriately. This is because the \texttt{oneSentencePerLine} has been instructed to store sentences (because Listing 408); each sentence is then searched for code blocks.

\textbf{example 116}

We can tweak the settings in Listing 376 on page 101 to ensure that full stops are not followed by \texttt{item} commands, and that the end of sentences contains \texttt{\end{itemize}} as in Listing 412. This setting is actually an appended version of the \texttt{betterFullStop} from the \texttt{fineTuning}, detailed in Listing 567 on page 147.
6.2 oneSentencePerLine: modifying line breaks for sentences

Upon running

```
cmh:~$ latexindent.pl multiple-sentences6 -m -l=sentence-wrap1.yaml,itemize.yaml
```

we receive the output in Listing 413.

Notice that the sentence has received indentation, and that the `itemize` code block has been found and indented correctly.

Text wrapping when using the `oneSentencePerLine` routine determines if it will remove line breaks while text wrapping, from the value of `removeSentenceLineBreaks`. 

---

**LISTING 412: itemize.yaml**

```yaml
modifyLineBreaks:
  textWrapOptions:
    columns: 45
  oneSentencePerLine:
    sentencesEndWith:
      betterFullStop: 0
      other: |
        (?x)
        (?:
          (?:\R|\h)*\item
        ) # new
        |
        (?:
          \.
        ) # new
        (?\h*[a-z])
        |
        (?<!
          (?:
            (?:[eE][gG])
          ) # new
          (?:[iI][eE])
          | # new
          (?:etc)
        )
        |
        \.
        (?:\h*R(?::\end\{itemize\})?) # new
        (?!
          (?:
            [a-zA-Z0-9-.,]
          )
          | # new
          \,
          | # new
        )
      |
```

---

**LISTING 413: multiple-sentences6-mod3.tex**

Consider the following:

```
\begin{itemize}
  \item firstly.
  \item secondly.
\end{itemize}
```

Notice that the sentence has received indentation, and that the `itemize` code block has been found and indented correctly.
6.2 oneSentencePerLine: text wrapping and indenting sentences, when before/after

The text wrapping routine operates, by default, before the code blocks have been found, but this can be changed to after:

- before means it is likely that the columns of wrapped text may exceed the value specified in columns;
- after means it columns of wrapped text should not exceed the value specified in columns.

We demonstrate this in the following examples. See also Section 6.1.7.

example 117

Let’s begin with the file in Listing 414.

\begin{myenv}
This paragraph has line breaks throughout its paragraph; we would like to combine the textwrapping and paragraph removal routine.
\end{myenv}

Using the settings given in Listing 416 and running the command

cmh:~$ latexindent.pl multiple-sentences8 -o=+-mod1.tex -l=sentence-wrap2 -m

gives the output given in Listing 415.

We note that, in Listing 415, that the wrapped text has exceeded the specified value of columns (35) given in Listing 416. We can affect this by changing when; we explore this next.

example 118

We continue working with Listing 414.

Using the settings given in Listing 418 and running the command

cmh:~$ latexindent.pl multiple-sentences8.tex -o=+-mod2.tex -l=sentence-wrap3 -m

gives the output given in Listing 417.
6.3 Poly-switches

Every other field in the modifyLineBreaks field uses poly-switches, and can take one of the following integer values:

- `defaultIndent`
- `modifyLineBreaks`
  - `oneSentencePerLine`
    - `manipulateSentences`
    - `textWrapSentences`
  - `textWrapOptions`
    - `columns`
    - `comments`
      - `wrap`

We note that, in Listing 420, that the sentences have been wrapped, and so too have the comments because of the annotated line in Listing 421.

We note that, in Listing 417, that the wrapped text has obeyed the specified value of `columns` (35) given in Listing 418.

6.2.9 oneSentencePerLine: text wrapping sentences and comments

We demonstrate the one sentence per line routine with respect to text wrapping comments. See also Section 6.1.8.

element 119

Let's begin with the file in Listing 419.

Using the settings given in Listing 421 and running the command

cmh:~$ latexindent.pl multiple-sentences9 -o=-mod1.tex -l=sentence-wrap4 -m

gives the output given in Listing 420.

We note that, in Listing 420, that the sentences have been wrapped, and so too have the comments because of the annotated line in Listing 421.
remove mode: line breaks before or after the \texttt{<part of thing>} can be removed (assuming that \texttt{preserveBlankLines} is set to 0);

off mode: line breaks will not be modified for the \texttt{<part of thing>} under consideration;

add mode: a line break will be added before or after the \texttt{<part of thing>} under consideration, assuming that there is not already a line break before or after the \texttt{<part of thing>};

comment then add mode: a comment symbol will be added, followed by a line break before or after the \texttt{<part of thing>} under consideration, assuming that there is not already a comment and line break before or after the \texttt{<part of thing>};

add then blank line mode: a line break will be added before or after the \texttt{<part of thing>} under consideration, assuming that there is not already a line break before or after the \texttt{<part of thing>}, followed by a blank line;

add blank line mode: a blank line will be added before or after the \texttt{<part of thing>} under consideration, even if the \texttt{<part of thing>} is already on its own line.

In the above, \texttt{<part of thing>} refers to either the \texttt{begin statement}, \texttt{body} or \texttt{end statement} of the code blocks detailed in Table 2 on page 57. All poly-switches are off by default; \texttt{latexindent.pl} searches first of all for per-name settings, and then followed by global per-thing settings.

### 6.3.1 Poly-switches for environments

We start by viewing a snippet of \texttt{defaultSettings.yaml} in Listing 422; note that it contains \texttt{global} settings (immediately after the \texttt{environments} field) and that \texttt{per-name} settings are also allowed – in the case of Listing 422, settings for \texttt{equation*} have been specified for demonstration. Note that all poly-switches are off (set to 0) by default.

```
environments:
  BeginStartsOnOwnLine: 0  # -1,0,1,2,3,4
  BodyStartsOnOwnLine: 0  # -1,0,1,2,3,4
  EndStartsOnOwnLine: 0   # -1,0,1,2,3,4
  EndFinishesWithLineBreak: 0 # -1,0,1,2,3,4

# equation*
# BeginStartsOnOwnLine: 0  # -1,0,1,2,3,4
# BodyStartsOnOwnLine: 0  # -1,0,1,2,3,4
# EndStartsOnOwnLine: 0   # -1,0,1,2,3,4
# EndFinishesWithLineBreak: 0 # -1,0,1,2,3,4
```

Let’s begin with the simple example given in Listing 423; note that we have annotated key parts of the file using ♠, ♥, ♦ and ♣, these will be related to fields specified in Listing 422.

```
\texttt{before words}♠ \begin{myenv}♥ \texttt{body of myenv}♦ \end{myenv}♣ \texttt{after words}
```

### 6.3.1.1 Adding line breaks: \texttt{BeginStartsOnOwnLine} and \texttt{BodyStartsOnOwnLine}

example 120

Let’s explore \texttt{BeginStartsOnOwnLine} and \texttt{BodyStartsOnOwnLine} in Listings 424 and 425, and in particular, let’s allow each of them in turn to take a value of 1.
After running the following commands,

```
cmh:~$ latexindent.pl -m env-mlb.tex -l env-mlb1.yaml
cmh:~$ latexindent.pl -m env-mlb.tex -l env-mlb2.yaml
```

the output is as in Listings 426 and 427 respectively.

Listing 426: env-mlb.tex using Listing 424

```
before words
\begin{myenv}body of myenv\end{myenv} after words
```

Listing 427: env-mlb.tex using Listing 425

```
before words \begin{myenv}body of myenv\end{myenv} after words
```

There are a couple of points to note:

- in Listing 426 a line break has been added at the point denoted by ♠ in Listing 423; no other line breaks have been changed;
- in Listing 427 a line break has been added at the point denoted by ♥ in Listing 423; furthermore, note that the body of myenv has received the appropriate (default) indentation.

**example 121**

Let’s now change each of the 1 values in Listings 424 and 425 so that they are 2 and save them into env-mlb3.yaml and env-mlb4.yaml respectively (see Listings 428 and 429).

Listing 428: env-mlb3.yaml

```
modifyLineBreaks:
  environments:
    BeginStartsOnOwnLine: 2
```

Listing 429: env-mlb4.yaml

```
modifyLineBreaks:
  environments:
    BodyStartsOnOwnLine: 2
```

Upon running the commands

```
cmh:~$ latexindent.pl -m env-mlb.tex -l env-mlb3.yaml
cmh:~$ latexindent.pl -m env-mlb.tex -l env-mlb4.yaml
```

we obtain Listings 430 and 431.

Listing 430: env-mlb.tex using Listing 428

```
before words%
\begin{myenv}body of myenv\end{myenv} after words
```

Listing 431: env-mlb.tex using Listing 429

```
before words \begin{myenv}body of myenv\end{myenv} after words
```

Note that line breaks have been added as in Listings 426 and 427, but this time a comment symbol has been added before adding the line break; in both cases, trailing horizontal space has been stripped before doing so.

**example 122**

Let’s now change each of the 1 values in Listings 424 and 425 so that they are 3 and save them into env-mlb5.yaml and env-mlb6.yaml respectively (see Listings 432 and 433).

Listing 432: env-mlb5.yaml

```
modifyLineBreaks:
  environments:
    BeginStartsOnOwnLine: 3
```

Listing 433: env-mlb6.yaml

```
modifyLineBreaks:
  environments:
    BodyStartsOnOwnLine: 3
```

Upon running the commands

```
cmh:~$ latexindent.pl -m env-mlb.tex -l env-mlb5.yaml
cmh:~$ latexindent.pl -m env-mlb.tex -l env-mlb6.yaml
```
we obtain Listings 434 and 435.

Note that line breaks have been added as in Listings 426 and 427, but this time a blank line has been added after adding the line break.

example 123

Let's now change each of the 1 values in Listings 432 and 433 so that they are 4 and save them into env-beg4.yaml and env-body4.yaml respectively (see Listings 436 and 437).

We will demonstrate this poly-switch value using the code in Listing 438.

Upon running the commands

```
cmh:~$ latexindent.pl -m env-mlb1.tex -l env-beg4.yaml
cmh:~$ latexindent.pl -m env-mlb1.tex -l env-body4.yaml
```

then we receive the respective outputs in Listings 439 and 440.

We note in particular that, by design, for this value of the poly-switches:

1. in Listing 439 a blank line has been inserted before the \begin statement, even though the \begin statement was already on its own line;
2. in Listing 440 a blank line has been inserted before the beginning of the body, even though it already began on its own line.

6.3.1.2 Adding line breaks: EndStartsOnOwnLine and EndFinishesWithLineBreak

example 124

Let's explore EndStartsOnOwnLine and EndFinishesWithLineBreak in Listings 441 and 442, and in particular, let's allow each of them in turn to take a value of 1.
After running the following commands,

```
cmh:~$ latexindent.pl -m env-lbl.tex -l env-mlb7.yaml
cmh:~$ latexindent.pl -m env-lbl.tex -l env-mlb8.yaml
```

the output is as in Listings 443 and 444.

There are a couple of points to note:

- in Listing 443 a line break has been added at the point denoted by ♦ in Listing 423 on page 112; no other line breaks have been changed and the `\end{myenv}` statement has not received indentation (as intended);
- in Listing 444 a line break has been added at the point denoted by ♣ in Listing 423 on page 112.

example 125

Let's now change each of the 1 values in Listings 441 and 442 so that they are 2 and save them into env-mlb9.yaml and env-mlb10.yaml respectively (see Listings 445 and 446).

Upon running the commands

```
cmh:~$ latexindent.pl -m env-lbl.tex -l env-mlb9.yaml
cmh:~$ latexindent.pl -m env-lbl.tex -l env-mlb10.yaml
```

we obtain Listings 447 and 448.

Note that line breaks have been added as in Listings 443 and 444, but this time a comment symbol has been added before adding the line break; in both cases, trailing horizontal space has been stripped before doing so.

example 126

Let's now change each of the 1 values in Listings 441 and 442 so that they are 3 and save them into env-mlb11.yaml and env-mlb12.yaml respectively (see Listings 449 and 450).
Upon running the commands

```bash
$ latexindent.pl -m env-mlb1.tex -l env-mlb11.yaml
$ latexindent.pl -m env-mlb2.tex -l env-mlb12.yaml
```

we obtain Listings 451 and 452.

Note that line breaks have been added as in Listings 443 and 444, and that a blank line has been added after the line break.

### example 127

Let’s now change each of the 1 values in Listings 449 and 450 so that they are 4 and save them into env-end4.yaml and env-end-f4.yaml respectively (see Listings 453 and 454).

We will demonstrate this poly-switch value using the code from Listing 438 on page 114.

Upon running the commands

```bash
$ latexindent.pl -m env-mlb1.tex -l env-end4.yaml
$ latexindent.pl -m env-mlb2.tex -l env-end-f4.yaml
```

then we receive the respective outputs in Listings 455 and 456.

We note in particular that, by design, for this value of the poly-switches:

1. in Listing 455 a blank line has been inserted before the `\end` statement, even though the `\end` statement was already on its own line;
2. in Listing 456 a blank line has been inserted after the `\end` statement, even though it already began on its own line.
6.3.1.3 poly-switches 1, 2, and 3 only add line breaks when necessary

If you ask \texttt{latexindent.pl} to add a line break (possibly with a comment) using a poly-switch value of 1 (or 2 or 3), it will only do so if necessary.

\textbf{example 128}

For example, if you process the file in Listing 457 using poly-switch values of 1, 2, or 3, it will be left unchanged.

\begin{verbatim}
before words \begin{myenv} body of myenv \end{myenv} after words
\end{verbatim}

Setting the poly-switches to a value of 4 instructs \texttt{latexindent.pl} to add a line break even if the \texttt{<part of thing>} is already on its own line; see Listings 439 and 440 and Listings 455 and 456.

\textbf{example 129}

In contrast, the output from processing the file in Listing 458 will vary depending on the poly-switches used; in Listing 459 you'll see that the comment symbol after the \texttt{\begin{myenv}} has been moved to the next line, as \texttt{BodyStartsOnOwnLine} is set to 1. In Listing 460 you'll see that the comment has been accounted for correctly because \texttt{BodyStartsOnOwnLine} has been set to 2, and the comment symbol has not been moved to its own line. You're encouraged to experiment with Listing 458 and by setting the other poly-switches considered so far to 2 in turn.

The details of the discussion in this section have concerned \textit{global} poly-switches in the \texttt{environments} field; each switch can also be specified on a \textit{per-name} basis, which would take priority over the global values; with reference to Listing 422 on page 112, an example is shown for the \texttt{equation*} environment.

6.3.1.4 Removing line breaks (poly-switches set to $-1$)

Setting poly-switches to $-1$ tells \texttt{latexindent.pl} to remove line breaks of the \texttt{<part of the thing>}, if necessary.

\textbf{example 130}

We will consider the example code given in Listing 461, noting in particular the positions of the line break highlighters, ♠, ♥, ♦ and ♣, together with the associated YAML files in Listings 462 to 465.
After running the commands

```bash
cmh:~$ latexindent.pl -m env-mlb13.yaml env-mlb4.tex -l
cmh:~$ latexindent.pl -m env-mlb14.yaml env-mlb4.tex -l
cmh:~$ latexindent.pl -m env-mlb15.yaml env-mlb4.tex -l
cmh:~$ latexindent.pl -m env-mlb16.yaml env-mlb4.tex -l
```

we obtain the respective output in Listings 466 to 469.

### Listing 466: env-mlb4.tex using Listing 462
```latex
\begin{myenv}
\text{body of myenv}
\end{myenv}
```

### Listing 467: env-mlb4.tex using Listing 463
```latex
\begin{myenv}
\text{body of myenv}
\end{myenv}
```

### Listing 468: env-mlb4.tex using Listing 464
```latex
\begin{myenv}
\text{body of myenv}
\end{myenv}
```

### Listing 469: env-mlb4.tex using Listing 465
```latex
\begin{myenv}
\text{body of myenv}
\end{myenv}
```

Notice that in:
- Listing 466 the line break denoted by ♠ in Listing 461 has been removed;
- Listing 467 the line break denoted by ♥ in Listing 461 has been removed;
- Listing 468 the line break denoted by ♦ in Listing 461 has been removed;
- Listing 469 the line break denoted by ♣ in Listing 461 has been removed.

We examined each of these cases separately for clarity of explanation, but you can combine all of the YAML settings in Listings 462 to 465 into one file; alternatively, you could tell `latexindent.pl` to load them all by using the following command, for example

```bash
cmh:~$ latexindent.pl -m env-mlb13.yaml -m env-mlb14.yaml -m env-mlb15.yaml -m env-mlb16.yaml env-mlb4.tex -l
```
6.3 Poly-switches

6.3.1.5 About trailing horizontal space

Recall that on page 33 we discussed the YAML field `removeTrailingWhitespace`, and that it has two (binary) switches to determine if horizontal space should be removed `beforeProcessing` and `afterProcessing`. The `beforeProcessing` is particularly relevant when considering the `-m` switch.

example 131

We consider the file shown in Listing 470, which highlights trailing spaces.

```latex
\begin{myenv}
body of myenv
\end{myenv}
```

The output from the following commands

```sh
cmh:~$ latexindent.pl -m env-mlb5.tex -l env-mlb13,env-mlb14,env-mlb15,env-mlb16
```

is shown, respectively, in Listings 472 and 473; note that the trailing horizontal white space has been preserved (by default) in Listing 472, while in Listing 473, it has been removed using the switch specified in Listing 471.

```yaml
modifyLineBreaks:
  preserveBlankLines: 0
```

6.3.1.6 poly-switch line break removal and blank lines

example 132

Now let's consider the file in Listing 474, which contains blank lines.

```latex
\begin{myenv}
body of myenv
\end{myenv}
```

```yaml
UnpreserveBlankLines.yaml
modifyLineBreaks:
  preserveBlankLines: 0
```
Upon running the following commands

```
cmh:~$ latexindent.pl -m env-mlb6.tex -l env-mlb13,env-mlb14,env-mlb15,env-mlb16
```

we receive the respective outputs in Listings 476 and 477. In Listing 476 we see that the multiple blank lines have each been condensed into one blank line, but that blank lines have not been removed by the poly-switches – this is because, by default, preserveBlankLines is set to 1. By contrast, in Listing 477, we have allowed the poly-switches to remove blank lines because, in Listing 475, we have set preserveBlankLines to 0.

---

**Example 133**

We can explore this further using the blank-line poly-switch value of 3; let’s use the file given in Listing 478.

```
\begin{one} one text \end{one} \begin{two} two text \end{two}
```

Upon running the following commands

```
cmh:~$ latexindent.pl -m env-mlb7.tex -l env-mlb12.yaml,env-mlb13.yaml
```

we receive the outputs given in Listings 479 and 480.

---

Notice that in:

- Listing 479 that \end{one} has added a blank line, because of the value of EndFinishesWithLineBreak in Listing 450 on page 116, and even though the line break ahead of \begin{two} should have been removed (because of BeginStartsOnOwnLine in Listing 462 on page 118), the blank line has been preserved by default;

- Listing 480, by contrast, has had the additional line-break removed, because of the settings in Listing 475.
6.3.2 Poly-switches for double backslash

With reference to `lookForAlignDelims` (see Listing 59 on page 34) you can specify poly-switches to dictate the line-break behaviour of double backslashes in environments (Listing 61 on page 34), commands (Listing 95 on page 41), or special code blocks (Listing 140 on page 48).  

Consider the code given in Listing 481.

```
\begin{tabular}{cc}
1 & 2
\\ 3 & 4
\end{tabular}
```

Referencing Listing 481:

- DBS stands for *double backslash*;
- line breaks ahead of the double backslash are annotated by ★, and are controlled by `DBSStartsOnOwnLine`;
- line breaks after the double backslash are annotated by □, and are controlled by `DBSFinishesWithLineBreak`.

Let’s explore each of these in turn.

### 6.3.2.1 Double backslash starts on own line

#### example 134

We explore `DBSStartsOnOwnLine` (★ in Listing 481); starting with the code in Listing 481, together with the YAML files given in Listing 483 and Listing 485 and running the following commands

```
cmh:~$ latexindent.pl -m tabular3.tex -l DBS1.yaml
cmh:~$ latexindent.pl -m tabular3.tex -l DBS2.yaml
```

then we receive the respective output given in Listing 482 and Listing 484.

```
\begin{tabular}{cc}
1 & 2
\\ 3 & 4
\end{tabular}
```

```
\begin{tabular}{cc}
1 & 2 \\
3 & 4
\end{tabular}
```

We note that

- Listing 483 specifies `DBSStartsOnOwnLine` for *every* environment (that is within `lookForAlignDelims`, Listing 62 on page 35); the double backslashes from Listing 481 have been moved to their own line in Listing 482;
- Listing 485 specifies `DBSStartsOnOwnLine` on a *per-name* basis for `tabular` (that is within `lookForAlignDelims`, Listing 62 on page 35); the double backslashes from Listing 481 have been moved to their own line in Listing 484;

---

6There is no longer any need for the code block to be specified within `lookForAlignDelims` for DBS poly-switches to activate.
have been moved to their own line in Listing 484, having added comment symbols before moving them.

example 135

We can combine DBS poly-switches with, for example, the `alignContentAfterDoubleBackSlash` in Section 5.5.6 on page 46.

For example, starting with the file Listing 486, and using the settings in Listings 131 and 133 on page 47 and running

```
cmh:~$ latexindent.pl -s -m -l alignContentAfterDBS1.yaml,DBS1.yaml tabular6.tex -o=+-mod1
cmh:~$ latexindent.pl -s -m -l alignContentAfterDBS2.yaml,DBS1.yaml tabular6.tex -o=+-mod2
```

gives the respective outputs shown in Listings 487 and 488.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{tabular}{cc}</td>
<td>\begin{tabular}{cc}</td>
<td>\begin{tabular}{cc}</td>
</tr>
<tr>
<td>1&amp;22\333&amp;4444\55555&amp;666666</td>
<td>1 &amp; 22 \ 333 &amp; 4444 \ 55555 &amp; 666666</td>
<td>1 &amp; 22 \ 333 &amp; 4444 \ 55555 &amp; 666666</td>
</tr>
<tr>
<td>\end{tabular}</td>
<td>\end{tabular}</td>
<td>\end{tabular}</td>
</tr>
</tbody>
</table>

We note that:

- in Listing 487 the content after the double back slash has been aligned;
- in Listing 488 we see that 3 spaces have been added after the double back slash.

6.3.2.2 Double backslash finishes with line break

example 136

Let’s now explore `DBSFinishesWithLineBreak` in Listing 481; starting with the code in Listing 481, together with the YAML files given in Listing 490 and Listing 492 and running the following commands

```
cmh:~$ latexindent.pl -m tabular3.tex -l DBS3.yaml
cmh:~$ latexindent.pl -m tabular3.tex -l DBS4.yaml
```

then we receive the respective output given in Listing 489 and Listing 491.

<table>
<thead>
<tr>
<th>Listing 489: tabular3.tex using Listing 490</th>
<th>Listing 490: DBS3.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{tabular}{cc}</td>
<td>modifyLineBreaks:</td>
</tr>
<tr>
<td>1 &amp; 2 \ 3 &amp; 4 \</td>
<td>environments:</td>
</tr>
<tr>
<td>\end{tabular}</td>
<td>DBSFinishesWithLineBreak: 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 491: tabular3.tex using Listing 492</th>
<th>Listing 492: DBS4.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{tabular}{cc}</td>
<td>modifyLineBreaks:</td>
</tr>
<tr>
<td>1 &amp; 2 \ 3 &amp; 4 \</td>
<td>environments:</td>
</tr>
<tr>
<td>\end{tabular}</td>
<td>tabular:</td>
</tr>
</tbody>
</table>

We note that
• Listing 490 specifies DBSFinishesWithLineBreak for every environment (that is within lookForAlignDelims, Listing 62 on page 35); the code following the double backslashes from Listing 481 has been moved to their own line in Listing 489;

• Listing 492 specifies DBSFinishesWithLineBreak on a per-name basis for tabular (that is within lookForAlignDelims, Listing 62 on page 35); the first double backslashes from Listing 481 have moved code following them to their own line in Listing 491, having added comment symbols before moving them; the final double backslashes have not added a line break as they are at the end of the body within the code block.

6.3.2.3 Double backslash poly-switches for specialBeginEnd

element 137

Let’s explore the double backslash poly-switches for code blocks within specialBeginEnd code blocks (Listing 138 on page 48); we begin with the code within Listing 493.

```
\< a & =b \ \\
 & =c \ \\
 & =d \ \\
 & =e \\
\>
```

Upon using the YAML settings in Listing 495, and running the command

```
cmh:~$ latexindent .pl -m special4.tex -l DBS5.yaml
```

then we receive the output given in Listing 494.

```
\<
 a & =b \\
 & =c \\
 & =d \\
 & =e \\
\>
```

There are a few things to note:

• in Listing 495 we have specified cmhMath within lookForAlignDelims; without this, the double backslash poly-switches would be ignored for this code block;

• the DBSFinishesWithLineBreak poly-switch has controlled the line breaks following the double backslashes;

• the SpecialEndStartsOnOwnLine poly-switch has controlled the addition of a comment symbol, followed by a line break, as it is set to a value of 2.

6.3.2.4 Double backslash poly-switches for optional and mandatory arguments

For clarity, we provide a demonstration of controlling the double backslash poly-switches for optional and mandatory arguments.

example 138

We use with the code in Listing 496.
6.3 Poly-switches

Upon using the YAML settings in Listings 498 and 500, and running the command

```
cmh:~$ latexindent.pl -m mycommand2.tex -l DBS6.yaml
cmh:~$ latexindent.pl -m mycommand2.tex -l DBS7.yaml
```
then we receive the output given in Listings 497 and 499.

**6.3.2.5 Double backslash optional square brackets**

The pattern matching for the double backslash will also, optionally, allow trailing square brackets that contain a measurement of vertical spacing, for example `\[3pt]`.

**Example 139**

For example, beginning with the code in Listing 501

```
\begin{pmatrix}
  1 & 2 \\
  3 & 4 \\
  5 & 6 \\
  7 & 8 \\
\end{pmatrix}
```

and running the following command, using Listing 490,

```
cmh:~$ latexindent.pl -m pmatrix3.tex -l DBS3.yaml
```
then we receive the output given in Listing 502.
6.3 Poly-switches

You can customise the pattern for the double backslash by exploring the fine tuning field detailed in Listing 567 on page 147.

6.3.3 Poly-switches for other code blocks

Rather than repeat the examples shown for the environment code blocks (in Section 6.3.1 on page 112), we choose to detail the poly-switches for all other code blocks in Table 3; note that each and every one of these poly-switches is off by default, i.e., set to 0.

Note also that, by design, line breaks involving, filecontents and 'comment-marked' code blocks (Listing 96 on page 41) can not be modified using latexindent.pl. However, there are two poly-switches available for verbatim code blocks: environments (Listing 38 on page 29), commands (Listing 39 on page 29) and specialBeginEnd (Listing 155 on page 52).
### Table 3: Poly-switch mappings for all code-block types

<table>
<thead>
<tr>
<th>Code block</th>
<th>Sample</th>
<th>Poly-switch mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>environment</td>
<td>before words♠</td>
<td>♠ BeginStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td>\begin{myenv}♥</td>
<td>♥ BodyStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td>body of myenv♦</td>
<td>♦ EndStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td>\end{myenv}♣</td>
<td>♣ EndFinishesWithLineBreak</td>
</tr>
<tr>
<td></td>
<td>after words</td>
<td></td>
</tr>
<tr>
<td>if/elsefi</td>
<td>before words♠</td>
<td>♠ IfStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td>\if...♥</td>
<td>♥ BodyStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td>body of if/or statement▲</td>
<td>▲ OrStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td>\or▼</td>
<td>▼ OrFinishesWithLineBreak</td>
</tr>
<tr>
<td></td>
<td>body of if/or statement★</td>
<td>★ ElseStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td>\else□</td>
<td>□ ElseFinishesWithLineBreak</td>
</tr>
<tr>
<td></td>
<td>body of else statement♦</td>
<td>♦ FiStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td>\fi♣</td>
<td>♣ FiFinishesWithLineBreak</td>
</tr>
<tr>
<td></td>
<td>after words</td>
<td></td>
</tr>
<tr>
<td>optionalArguments</td>
<td>...♠</td>
<td>♠ LSqBStartsOnOwnLine²</td>
</tr>
<tr>
<td></td>
<td>[♥</td>
<td>♥ OptArgBodyStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td>value before comma★,</td>
<td>★ CommaStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td>□</td>
<td>□ CommaFinishesWithLineBreak</td>
</tr>
<tr>
<td></td>
<td>end of body of opt arg♦</td>
<td>◆ RSqBStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td>]♣</td>
<td>♣ RSqBFinishesWithLineBreak</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>mandatoryArguments</td>
<td>...♠</td>
<td>♠ LCuBStartsOnOwnLine²</td>
</tr>
<tr>
<td></td>
<td>{♥</td>
<td>♥ MandArgBodyStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td>value before comma★,</td>
<td>★ CommaStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td>□</td>
<td>□ CommaFinishesWithLineBreak</td>
</tr>
<tr>
<td></td>
<td>end of body of mand arg♦</td>
<td>◆ RCuBStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td>}♣</td>
<td>♣ RCuBFinishesWithLineBreak</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>commands</td>
<td>before words♠</td>
<td>♠ CommandStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td>\mycommand♥</td>
<td>♥ CommandNameFinishesWithLineBreak</td>
</tr>
<tr>
<td></td>
<td>(arguments)</td>
<td></td>
</tr>
<tr>
<td>namedGroupingBracesBrackets</td>
<td>before words♠</td>
<td>♠ NameStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td>myname♥</td>
<td>♥ NameFinishesWithLineBreak</td>
</tr>
<tr>
<td></td>
<td>(braces/brackets)</td>
<td></td>
</tr>
<tr>
<td>keyEqualsValuesBracesBrackets</td>
<td>before words♠</td>
<td>♠ KeyStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td>key♥</td>
<td>♥ EqualsStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td>(braces/brackets)</td>
<td></td>
</tr>
<tr>
<td>items</td>
<td>before words♠</td>
<td>♠ ItemStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td>\item♥</td>
<td>♥ ItemFinishesWithLineBreak</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>specialBeginEnd</td>
<td>before words♠</td>
<td>♠ SpecialBeginStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td>[^♥</td>
<td>♥ SpecialBodyStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td>body of special/middle★</td>
<td>★ SpecialMiddleStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td>\middle□</td>
<td>□ SpecialMiddleFinishesWithLineBreak</td>
</tr>
<tr>
<td></td>
<td>body of special/middle◊</td>
<td>◆ SpecialEndStartsOnOwnLine</td>
</tr>
<tr>
<td></td>
<td>]♣</td>
<td>♣ SpecialEndFinishesWithLineBreak</td>
</tr>
<tr>
<td></td>
<td>after words</td>
<td></td>
</tr>
<tr>
<td>verbatim</td>
<td>before words♠ \begin{verbatim}♣</td>
<td>♠ VerbatimBeginStartsOnOwnLine</td>
</tr>
</tbody>
</table>

²LSqB stands for Left Square Bracket

³LCuB stands for Left Curly Brace
6.3 Poly-switches

6.3.4 Partnering BodyStartsOnOwnLine with argument-based poly-switches

Some poly-switches need to be partnered together; in particular, when line breaks involving the first argument of a code block need to be accounted for using both BodyStartsOnOwnLine (or its equivalent, see Table 3 on the previous page) and LCuBStartsOnOwnLine for mandatory arguments, and LSqBStartsOnOwnLine for optional arguments.

example 140

Let’s begin with the code in Listing 503 and the YAML settings in Listing 505; with reference to Table 3 on the preceding page, the key CommandNameFinishesWithLineBreak is an alias for BodyStartsOnOwnLine.

```
\mycommand{
  mand arg text
  mand arg text}
  mand arg text
  mand arg text}
```

Listing 503: mycommand1.tex

Upon running the command

```
$ latexindent.pl -m -l=mycom-mlb1.yaml mycommand1.tex
```

we obtain Listing 504; note that the second mandatory argument beginning brace \{ has had its leading line break removed, but that the first brace has not.

```
\mycommand{
  mand arg text
  mand arg text}{
  mand arg text
  mand arg text}
```

Listing 504: mycommand1.tex using Listing 505

Listing 505: mycom-mlb1.yaml

```
modifyLineBreaks:
  commands:
    CommandNameFinishesWithLineBreak: 0
  mandatoryArguments:
    LCuBStartsOnOwnLine: -1
```

example 141

Now let’s change the YAML file so that it is as in Listing 507; upon running the command

```
$ latexindent.pl -m -l=mycom-mlb2.yaml mycommand1.tex
```

we obtain Listing 506; both beginning braces \{ have had their leading line breaks removed.

```
\mycommand{
  mand arg text
  mand arg text}{
  mand arg text
  mand arg text}
```

Listing 506: mycommand1.tex using Listing 507

Listing 507: mycom-mlb2.yaml

```
modifyLineBreaks:
  commands:
    CommandNameFinishesWithLineBreak: -1
  mandatoryArguments:
    LCuBStartsOnOwnLine: -1
```
6.3 Poly-switches

example 142

Now let's change the YAML file so that it is as in Listing 509; upon running the command

```bash
cmh:~$ latexindent.pl -m -l=mycom-mlb3.yaml mycommand1.tex
```

we obtain Listing 508.

<table>
<thead>
<tr>
<th>Listing 508: mycommand1.tex using Listing 509</th>
</tr>
</thead>
<tbody>
<tr>
<td>\mycommand</td>
</tr>
<tr>
<td>{</td>
</tr>
<tr>
<td>\mand \arg text</td>
</tr>
<tr>
<td>\mand \arg text</td>
</tr>
<tr>
<td>}</td>
</tr>
<tr>
<td>{</td>
</tr>
<tr>
<td>\mand \arg text</td>
</tr>
<tr>
<td>\mand \arg text</td>
</tr>
<tr>
<td>}</td>
</tr>
</tbody>
</table>

| Listing 509: mycom-mlb3.yaml                  |
| modifyLineBreaks:                             |
| commands:                                     |
|  CommandNameFinishesWithLineBreak: -1         |
| mandatoryArguments:                           |
|  LCuBStartsOnOwnLine: 1                       |

6.3.5 Conflicting poly-switches: sequential code blocks

It is very easy to have conflicting poly-switches.

example 143

We use the example from Listing 503 on the previous page, and consider the YAML settings given in Listing 511. The output from running

```bash
cmh:~$ latexindent.pl -m -l=mycom-mlb4.yaml mycommand1.tex
```

is given in Listing 511.

<table>
<thead>
<tr>
<th>Listing 510: mycommand1.tex using Listing 511</th>
</tr>
</thead>
<tbody>
<tr>
<td>\mycommand</td>
</tr>
<tr>
<td>{</td>
</tr>
<tr>
<td>\mand \arg text</td>
</tr>
<tr>
<td>\mand \arg text</td>
</tr>
<tr>
<td>}</td>
</tr>
<tr>
<td>{</td>
</tr>
<tr>
<td>\mand \arg text</td>
</tr>
<tr>
<td>\mand \arg text</td>
</tr>
<tr>
<td>}</td>
</tr>
</tbody>
</table>

Studying Listing 511, we see that the two poly-switches are at opposition with one another:

- on the one hand, LCuBStartsOnOwnLine should not start on its own line (as poly-switch is set to −1);
- on the other hand, RCuBFinishesWithLineBreak should finish with a line break.

So, which should win the conflict? As demonstrated in Listing 510, it is clear that LCuBStartsOnOwnLine won this conflict, and the reason is that the second argument was processed after the first – in general, the most recently-processed code block and associated poly-switch takes priority.

example 144

We can explore this further by considering the YAML settings in Listing 513; upon running the command

```bash
cmh:~$ latexindent.pl -m -l=mycom-mlb5.yaml mycommand1.tex
```

we obtain the output given in Listing 512.
As previously, the most-recently-processed code block takes priority – as before, the second (i.e., last) argument.

Exploring this further, we consider the YAML settings in Listing 515, and run the command

```
cmh:~$ latexindent.pl -m -l=mycom-mlb6.yaml mycommand1.tex
```

which gives the output in Listing 514.

Note that a `% has been added to the trailing first `); this is because:

- while processing the first argument, the trailing line break has been removed (RCuBFinishesWithLineBreak set to −1);
- while processing the second argument, latexindent.pl finds that it does not begin on its own line, and so because LCuBStartsOnOwnLine is set to 2, it adds a comment, followed by a line break.

### 6.3.6 Conflicting poly-switches: nested code blocks

**example 145**

Now let's consider an example when nested code blocks have conflicting poly-switches; we'll use the code in Listing 516, noting that it contains nested environments.

```
\begin{one}
one text
\begin{two}
two text
\end{two}
\end{one}
```

Let's use the YAML settings given in Listing 518, which upon running the command

```
cmh:~$ latexindent.pl -m -l=nested-env-mlb1.yaml nested-env.tex
```

gives the output in Listing 517.
In Listing 517, let’s first of all note that both environments have received the appropriate (default) indentation; secondly, note that the poly-switch EndStartsOnOwnLine appears to have won the conflict, as \end{one} has had its leading line break removed.

To understand it, let’s talk about the three basic phases of \texttt{latexindent.pl}:

1. Phase 1: packing, in which code blocks are replaced with unique ids, working from \textit{the inside to the outside}, and then sequentially – for example, in Listing 516, the \texttt{two} environment is found \textit{before} the \texttt{one} environment; if the -m switch is active, then during this phase:
   - line breaks at the beginning of the body can be added (if BodyStartsOnOwnLine is 1 or 2) or removed (if BodyStartsOnOwnLine is \texttt{-1});
   - line breaks at the end of the body can be added (if EndStartsOnOwnLine is 1 or 2) or removed (if EndStartsOnOwnLine is \texttt{-1});
   - line breaks after the end statement can be added (if EndFinishesWithLineBreak is 1 or 2).

2. Phase 2: indentation, in which white space is added to the begin, body, and end statements;

3. Phase 3: unpacking, in which unique ids are replaced by their \textit{indented} code blocks; if the -m switch is active, then during this phase,
   - line breaks before begin statements can be added or removed (depending upon BeginStartsOnOwnLine);
   - line breaks after end statements can be removed but \textbf{NOT} added (see EndFinishesWithLineBreak).

With reference to Listing 517, this means that during Phase 1:

- the \texttt{two} environment is found first, and the line break ahead of the \texttt{\end{two}} statement is removed because EndStartsOnOwnLine is set to \texttt{-1}. Importantly, because, at this stage, \texttt{\end{two}} does finish with a line break, EndFinishesWithLineBreak causes no action.
- next, the \texttt{one} environment is found; the line break ahead of \texttt{\end{one}} is removed because EndStartsOnOwnLine is set to \texttt{-1}.

The indentation is done in Phase 2; in Phase 3 there is no option to add a line break after the end statements. We can justify this by remembering that during Phase 3, the \texttt{one} environment will be found and processed first, followed by the \texttt{two} environment. If the \texttt{two} environment were to add a line break after the \texttt{\end{two}} statement, then \texttt{latexindent.pl} would have no way of knowing how much indentation to add to the subsequent text (in this case, \texttt{\end{one}}).

\example{146}{We can explore this further using the poly-switches in Listing 520; upon running the command}

\begin{verbatim}
 cmh:~$ latexindent.pl -m -l=nested-env-mlb2.yaml nested-env.tex
\end{verbatim}

we obtain the output given in Listing 519.
6.3 Poly-switches

LISTING 519: nested-env.tex using Listing 520

\begin{one}
  one text
\begin{two}
  two text
  \end{two}
\end{one}

During Phase 1:

- the \texttt{two} environment is found first, and the line break ahead of the \texttt{\end{two}} statement is not changed because \texttt{EndStartsOnOwnLine} is set to 1. Importantly, because, \emph{at this stage}, \texttt{\end{two}} does finish with a line break, \texttt{EndFinishesWithLineBreak} causes no action.

- next, the \texttt{one} environment is found; the line break ahead of \texttt{\end{one}} is already present, and no action is needed.

The indentation is done in Phase 2, and then in Phase 3, the \texttt{one} environment is found and processed first, followed by the \texttt{two} environment. \emph{At this stage}, the \texttt{two} environment finds \texttt{EndFinishesWithLineBreak} is \texttt{-1}, so it removes the trailing line break; remember, at this point, \texttt{latexindent.pl} has completely finished with the \texttt{one} environment.
SECTION 7

The -r, -rv and -rr switches

You can instruct `latexindent.pl` to perform replacements/substitutions on your file by using any of the -r, -rv or -rr switches:

- the -r switch will perform indentation and replacements, not respecting verbatim code blocks;
- the -rv switch will perform indentation and replacements, and will respect verbatim code blocks;
- the -rr switch will not perform indentation, and will perform replacements not respecting verbatim code blocks.

We will demonstrate each of the -r, -rv and -rr switches, but a summary is given in Table 4.

<table>
<thead>
<tr>
<th>Switch</th>
<th>Indentation?</th>
<th>Respect Verbatim?</th>
</tr>
</thead>
<tbody>
<tr>
<td>-r</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>-rv</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>-rr</td>
<td>×</td>
<td>×</td>
</tr>
</tbody>
</table>

The default value of the `replacements` field is shown in Listing 521; as with all of the other fields, you are encouraged to customise and change this as you see fit. The options in this field will only be considered if the -r, -rv or -rr switches are active; when discussing YAML settings related to the replacement-mode switches, we will use the style given in Listing 521.

```yaml
LISTING 521: replacements

replacements:
    - amalgamate: 1
    - this: latexindent.pl
      that: pl.latexindent
    lookForThis: 0
    when: before
```

The first entry within the `replacements` field is `amalgamate`, and is optional; by default it is set to 1, so that replacements will be amalgamated from each settings file that you specify. As you'll see in the demonstrations that follow, there is no need to specify this field.

You'll notice that, by default, there is only one entry in the `replacements` field, but it can take as many entries as you would like; each one needs to begin with a `-` on its own line.

### 7.1 Introduction to replacements

Let's explore the action of the default settings, and then we'll demonstrate the feature with further examples.

**example 147**

Beginning with the code in Listing 522 and running the command

```
cmh:~$ latexindent.pl -r replacel.tex
```
7.2 The two types of replacements

There are two types of replacements:

1. *string*-based replacements, which replace the string in *this* with the string in *that*. If you specify *this* and you do not specify *that*, then the *that* field will be assumed to be empty.

2. *regex*-based replacements, which use the *substitution* field.

We will demonstrate both in the examples that follow.

latexindent.pl chooses which type of replacement to make based on which fields have been specified; if the *this* field is specified, then it will make *string*-based replacements, regardless of if *substitution* is present or not.

7.3 Examples of replacements

element 148

We begin with code given in Listing 526

```
\begin{env}
1 2 3 \arraycolsep=3pt
4 5 6 \arraycolsep=5pt
\end{env}
```

Let’s assume that our goal is to remove both of the \arraycolsep statements; we can achieve this in a few different ways.
7.3 Examples of replacements

Using the YAML in Listing 528, and running the command

```
$ latexindent.pl -r colsep.tex -l=colsep.yaml
```

then we achieve the output in Listing 527.

Note that in Listing 528, we have specified two separate fields, each with their own 'this' field; furthermore, for both of the separate fields, we have not specified 'that', so the that field is assumed to be blank by latexindent.pl.

We can make the YAML in Listing 528 more concise by exploring the substitution field. Using the settings in Listing 530 and running the command

```
$ latexindent.pl -r colsep.tex -l=colsep1.yaml
```

then we achieve the output in Listing 529.

The code given in Listing 530 is an example of a regular expression, which we may abbreviate to regex in what follows. This manual is not intended to be a tutorial on regular expressions; you might like to read, for example, [34] for a detailed covering of the topic. With reference to Listing 530, we do note the following:

- the general form of the substitution field is `s/regex/replacement/modifiers`. You can place any regular expression you like within this;
- we have ‘escaped’ the backslash by using `\`
- we have used `\d+` to represent at least one digit
- the `s` modifier (in the `sg` at the end of the line) instructs `latexindent.pl` to treat your file as one single line;
- the `g` modifier (in the `sg` at the end of the line) instructs `latexindent.pl` to make the substitution globally throughout your file; you might try removing the `g` modifier from Listing 530 and observing the difference in output.

You might like to see https://perldoc.perl.org/perlre.html#Modifiers for details of modifiers; in general, I recommend starting with the `sg` modifiers for this feature.

**example 149**

We'll keep working with the file in Listing 526 on the previous page for this example.

Using the YAML in Listing 532, and running the command
7.3 Examples of replacements

\texttt{cmh:~}\$ \texttt{latexindent.pl -r colsep.tex -l=multi-line.yaml}

then we achieve the output in Listing 531.

\begin{Verbatim}
\textcolor{green}{\texttt{LISTING 531: colsep.tex using}}
\textcolor{green}{\texttt{Listing 532}}
\end{Verbatim}
\begin{Verbatim}
multi-line!
\end{Verbatim}

\begin{Verbatim}
\textcolor{green}{\texttt{LISTING 532: multi-line.yaml}}
\end{Verbatim}
\begin{Verbatim}
replacements:
- this: \texttt{|-}
  \begin{env}
  1 2 3\arraycolsep=3pt
  4 5 6\arraycolsep=5pt
  \end{env}
that: \texttt{\textquotesingle multi-line!\textquotesingle}
\end{Verbatim}

With reference to Listing 532, we have specified a \textit{multi-line} version of this by employing the \textit{literal} YAML style \texttt{|-}. See, for example, \url{https://stackoverflow.com/questions/3790454/in-yaml-how-do-i-break-a-string-over-multiple-lines} for further options, all of which can be used in your YAML file.

This is a natural point to explore the \texttt{when} field, specified in Listing 521 on page 132. This field can take two values: \texttt{before} and \texttt{after}, which respectively instruct \texttt{latexindent.pl} to perform the replacements \texttt{before} indentation or \texttt{after} it. The default value is \texttt{before}.

Using the YAML in Listing 534, and running the command

\texttt{cmh:~}\$ \texttt{latexindent.pl -r colsep.tex -l=multi-line1.yaml}

then we achieve the output in Listing 533.

\begin{Verbatim}
\textcolor{green}{\texttt{LISTING 533: colsep.tex using}}
\textcolor{green}{\texttt{Listing 534}}
\end{Verbatim}
\begin{Verbatim}
\begin{env}
  1 2 3\arraycolsep=3pt
  4 5 6\arraycolsep=5pt
\end{env}
\end{Verbatim}

\begin{Verbatim}
\textcolor{green}{\texttt{LISTING 534: multi-line1.yaml}}
\end{Verbatim}
\begin{Verbatim}
replacements:
- this: \texttt{|-}
  \begin{env}
  1 2 3\arraycolsep=3pt
  4 5 6\arraycolsep=5pt
  \end{env}
that: \texttt{\textquotesingle multi-line!\textquotesingle}
when: \texttt{after}
\end{Verbatim}

We note that, because we have specified \texttt{when: after}, that \texttt{latexindent.pl} has not found the string specified in Listing 534 within the file in Listing 526 on page 133. As it has looked for the string within Listing 534 \texttt{after} the indentation has been performed. After indentation, the string as written in Listing 534 is no longer part of the file, and has therefore not been replaced.

As a final note on this example, if you use the \texttt{-rr} switch, as follows,

\texttt{cmh:~}\$ \texttt{latexindent.pl -rr colsep.tex -l=multi-line1.yaml}

then the \texttt{when} field is ignored, no indentation is done, and the output is as in Listing 531.

\textbf{example 150}

An important part of the substitution routine is in \textit{capture groups}.

Assuming that we start with the code in Listing 535, let's assume that our goal is to replace each occurrence of \texttt{$\ldots$} with \texttt{\begin{equation*}\ldots\end{equation*}}. This example is partly motivated by \url{tex stackexchange question 242150}. 


7.3 Examples of replacements

<table>
<thead>
<tr>
<th>LISTING 535: displaymath.tex</th>
</tr>
</thead>
</table>
| before text $$a^2+b^2=4$$ and $$c^2$$

$$
\begin{align}
d^{2}+e^{2} &= f^{2} \\
\text{and also } g^{2} \\
\text{and some inline math: } h^{2}
\end{align}
$$

We use the settings in Listing 537 and run the command

```
$ latexindent.pl -r displaymath.tex -l=displaymath1.yaml
```

to receive the output given in Listing 536.

<table>
<thead>
<tr>
<th>LISTING 536: displaymath.tex using Listing 537</th>
</tr>
</thead>
</table>
| before text \begin{equation*}a^2+b^2=4\end{equation*} and \begin{equation*}c^2\end{equation*} \begin{equation*}d^2+e^2 = f^2\end{equation*} and also \begin{equation*}g^2\end{equation*} and some inline math: $h^{2}$

A few notes about Listing 537:

1. we have used the x modifier, which allows us to have white space within the regex;
2. we have used a capture group, (.*) which captures the content between the $$...$$ into the special variable, $1$;
3. we have used the content of the capture group, $1$, in the replacement text.

See [https://perldoc.perl.org/perlre.html#Capture-groups](https://perldoc.perl.org/perlre.html#Capture-groups) for a discussion of capture groups.

The features of the replacement switches can, of course, be combined with others from the toolkit of latexindent.pl. For example, we can combine the poly-switches of Section 6.3 on page 111, which we do in Listing 539; upon running the command

```
$ latexindent.pl -r -m displaymath.tex -l=displaymath1.yaml,equation.yaml
```

then we receive the output in Listing 538.
7.3 Examples of replacements

Listing 538: displaymath.tex using Listings 537 and 539

before text
\begin{equation} a^2+b^2=4 \end{equation}
and
\begin{equation} c^2 \end{equation}
\begin{equation} d^2+e^2 = f^2 \end{equation}
and also
\begin{equation} g^2 \end{equation}
and some inline math: \$h^2\$

example 151

This example is motivated by tex stackexchange question 490086. We begin with the code in Listing 540.

Listing 540: phrase.tex

| phrase 1 | phrase 2 | phrase 3 | phrase 100 |
| phrase 1 | phrase 2 | phrase 3 | phrase 100 |
| phrase 1 | phrase 2 | phrase 3 | phrase 100 |
| phrase 1 | phrase 2 | phrase 3 | phrase 100 |

Our goal is to make the spacing uniform between the phrases. To achieve this, we employ the settings in Listing 542, and run the command

```
example 151
```

which gives the output in Listing 541.

Listing 541: phrase.tex using Listing 542

| phrase 1 | phrase 2 | phrase 3 | phrase 100 |
| phrase 1 | phrase 2 | phrase 3 | phrase 100 |
| phrase 1 | phrase 2 | phrase 3 | phrase 100 |
| phrase 1 | phrase 2 | phrase 3 | phrase 100 |

The \h+ setting in Listing 542 say to replace at least one horizontal space with a single space.

example 152

We begin with the code in Listing 543.
Our goal is to change each reference so that both the text and the reference are contained within one hyperlink. We achieve this by employing Listing 545 and running the command

```
cmh:~$ latexindent.pl -r references.tex -l=reference.yaml
```

which gives the output in Listing 544.

Listing 544: references.tex using Listing 545

\hyperref{equation \ref*{eq:aa}} and \hyperref{Figure \ref*{fig:bb}} and \hyperref{table \ref*{tab:cc}}

Listing 545: reference.yaml

```
replacements:
- substitution: |-
  s/(equation | table | figure | section )<=(?:eq)?ref\{(.*?)\}/|\hyperref{$1 \ref*{$3}}/sgxi
```

Referencing Listing 545, the | means or, we have used capture groups, together with an example of an optional pattern, (?:eq)?.

**example 153**

Let's explore the three replacement mode switches (see Table 4 on page 132) in the context of an example that contains a verbatim code block, Listing 546; we will use the settings in Listing 547.

Listing 546: verb1.tex

```
\begin{myenv}
body of verbatim
\end{myenv}
some verbatim
\begin{verbatim}
body of verbatim text
\end{verbatim}
```

Listing 547: verbatim1.yaml

```
replacements:
- this: 'body'
  that: 'head'
```

Upon running the following commands,
7.3 Examples of replacements

We receive the respective output in Listings 548 to 550

**LISTING 548: verb1-mod1.tex**

```latex
\begin{myenv}
  head of verbatim
\end{myenv}
some verbatim
\begin{verbatim}
  head of verbatim
text
\end{verbatim}
text
```

**LISTING 549: verb1-rv-mod1.tex**

```latex
\begin{myenv}
  head of verbatim
\end{myenv}
\begin{verbatim}
  body of verbatim
text
\end{verbatim}
text
```

**LISTING 550: verb1-rr-mod1.tex**

```latex
\begin{myenv}
  head of verbatim
\end{myenv}
\begin{verbatim}
  head of verbatim
text
\end{verbatim}
text
```

We note that:

1. in Listing 548 indentation has been performed, and that the replacements specified in Listing 547 have been performed, even within the verbatim code block;

2. in Listing 549 indentation has been performed, but that the replacements have not been performed within the verbatim environment, because the rv switch is active;

3. in Listing 550 indentation has not been performed, but that replacements have been performed, not respecting the verbatim code block.

See the summary within Table 4 on page 132.

**Example 154**

Let’s explore the amalgamate field from Listing 521 on page 132 in the context of the file specified in Listing 551.

**LISTING 551: amalg1.tex**

```latex
one two three
```

Let’s consider the YAML files given in Listings 552 to 554.

**LISTING 552: amalg1-yaml.yaml**

```yaml
replacements:
  - this: one
    that: 1
```

**LISTING 553: amalg2-yaml.yaml**

```yaml
replacements:
  - this: two
    that: 2
```

**LISTING 554: amalg3-yaml.yaml**

```yaml
replacements:
  - amalgamate: 0
    this: three
    that: 3
```

Upon running the following commands,

```bash
$ latexindent.pl -r amalg1.tex -l=amalg1-yaml
$ latexindent.pl -r amalg1.tex -l=amalg1-yaml,amalg2-yaml
$ latexindent.pl -r amalg1.tex -l=amalg1-yaml,amalg2-yaml,amalg3-yaml
```

we receive the respective output in Listings 555 to 557.
We note that:

1. in Listing 555 the replacements from Listing 552 have been used;
2. in Listing 556 the replacements from Listings 552 and 553 have both been used, because the default value of `amalgamate` is 1;
3. in Listing 557 only the replacements from Listing 554 have been used, because the value of `amalgamate` has been set to 0.
SECTION 8

The –lines switch

\latexindent.pl can operate on a selection of lines of the file using the –lines or -n switch. The basic syntax is –lines MIN-MAX, so for example

\begin{verbatim}
cmh:~$ latexindent.pl --lines 3-7 myfile.tex
cmh:~$ latexindent.pl -n 3-7 myfile.tex
\end{verbatim}

will only operate upon lines 3 to 7 in myfile.tex. All of the other lines will not be operated upon by latexindent.pl.

The options for the lines switch are:

- line range, as in –lines 3-7
- single line, as in –lines 5
- multiple line ranges separated by commas, as in –lines 3-5,8-10
- negated line ranges, as in –lines !3-5 which translates to –lines 1-2,6-N, where N is the number of lines in your file.

We demonstrate this feature, and the available variations in what follows. We will use the file in Listing 558.

\begin{lstlisting}[caption=Listing 558: myfile.tex, label=lst:myfile] Before the environments
\begin{one}
  first block, first line
  first block, second line
  first block, third line
\end{one}
\begin{two}
  second block, first line
  second block, second line
  second block, third line
  second block, fourth line
\end{two}
\end{lstlisting}

\begin{example}[155]
We demonstrate the basic usage using the command

\begin{verbatim}
cmh:~$ latexindent.pl --lines 3-7 myfile.tex -o=-mod1
\end{verbatim}

which instructs latexindent.pl to only operate on lines 3 to 7; the output is given in Listing 559.
Before the environments
\begin{one}
first block, first line
first block, second line
first block, third line
\begin{two}
second block, first line
second block, second line
second block, third line
second block, fourth line
\end{two}
\end{one}

The following two calls to latexindent.pl are equivalent

cmh:~$ latexindent.pl --lines 3-7 myfile.tex -o=+-mod1
cmh:~$ latexindent.pl --lines 7-3 myfile.tex -o=+-mod1

as latexindent.pl performs a check to put the lowest number first.

example 156

You can call the lines switch with only one number and in which case only that line will be operated upon. For example

cmh:~$ latexindent.pl --lines 5 myfile.tex

cmh:~$ latexindent.pl --lines 5-5 myfile.tex

instructs latexindent.pl to only operate on line 5; the output is given in Listing 560.

LISTING 560: myfile-mod2.tex

Before the environments
\begin{one}
first block, first line
first block, second line
first block, third line
\begin{two}
second block, first line
second block, second line
second block, third line
second block, fourth line
\end{two}
\end{one}

The following two calls are equivalent:

cmh:~$ latexindent.pl --lines 5 myfile.tex
cmh:~$ latexindent.pl --lines 5-5 myfile.tex

example 157

If you specify a value outside of the line range of the file then latexindent.pl will ignore the lines argument, detail as such in the log file, and proceed to operate on the entire file.

For example, in the following call
latexindent.pl will ignore the lines argument, and operate on the entire file because Listing 558 only has 12 lines.

Similarly, in the call

```bash
cmh:~$ latexindent.pl --lines -1-3 myfile.tex
```

latexindent.pl will ignore the lines argument, and operate on the entire file because we assume that negatively numbered lines in a file do not exist.

**Example 158**

You can specify multiple line ranges as in the following

```bash
cmh:~$ latexindent.pl --lines 3-5,8-10 myfile.tex -o=+-mod3
```

which instructs latexindent.pl to operate upon lines 3 to 5 and lines 8 to 10; the output is given in Listing 561.

**Listing 561: myfile-mod3.tex**

```
1 Before the environments
2 \begin{one}
3 first block, first line
4 first block, second line
5 first block, third line
6 \begin{two}
7 second block, first line
8 second block, second line
9 second block, third line
10 second block, fourth line
11 \end{two}
12 \end{one}
```

The following calls to latexindent.pl are all equivalent

```bash
cmh:~$ latexindent.pl --lines 3-5,8-10 myfile.tex
```

as latexindent.pl performs a check to put the lowest line ranges first, and within each line range, it puts the lowest number first.

**Example 159**

There's no limit to the number of line ranges that you can specify, they just need to be separated by commas. For example

```bash
cmh:~$ latexindent.pl --lines 1-2,4-5,9-10,12 myfile.tex -o=+-mod4
```

has four line ranges: lines 1 to 2, lines 4 to 5, lines 9 to 10 and line 12. The output is given in Listing 562.
As previously, the ordering does not matter, and the following calls to `latexindent.pl` are all equivalent:

```bash
$ latexindent.pl --lines 1-2,4-5,9-10,12 myfile.tex
$ latexindent.pl --lines 2-1,4-5,9-10,12 myfile.tex
$ latexindent.pl --lines 4-5,1-2,9-10,12 myfile.tex
$ latexindent.pl --lines 12,4-5,1-2,9-10 myfile.tex
```

as `latexindent.pl` performs a check to put the lowest line ranges first, and within each line range, it puts the lowest number first.

**example 160**

You can specify *negated line ranges* by using `!` as in:

```bash
$ latexindent.pl --lines !5-7 myfile.tex -o=+-mod5
```

which instructs `latexindent.pl` to operate upon all of the lines except lines 5 to 7.

In other words, `latexindent.pl` will operate on lines 1 to 4, and 8 to 12, so the following two calls are equivalent:

```bash
$ latexindent.pl --lines !5-7 myfile.tex
$ latexindent.pl --lines 1-4,8-12 myfile.tex
```

The output is given in Listing 563.
example 161

You can specify _multiple negated line ranges_ such as

```bash
$ latexindent.pl --lines 5-7,9-10 myfile.tex -o=+-mod6
```

which is equivalent to:

```bash
$ latexindent.pl --lines 1-4,8,11-12 myfile.tex -o=+-mod6
```

The output is given in Listing 564.

**Listing 564: myfile-mod6.tex**

```latex
Before the environments
\begin{one}
  first block, first line
  first block, second line
  first block, third line
\end{one}
\begin{two}
  second block, first line
  second block, second line
  second block, third line
  second block, fourth line
\end{two}
\end{one}
```

example 162

If you specify a line range with anything other than an integer, then `latexindent.pl` will ignore the `lines` argument, and _operate on the entire file._

Sample calls that result in the `lines` argument being ignored include the following:

```bash
$ latexindent.pl --lines 1-x myfile.tex
$ latexindent.pl --lines !y-3 myfile.tex
```

example 163

We can, of course, use the `lines` switch in combination with other switches.

For example, let's use with the file in Listing 565.

**Listing 565: myfile1.tex**

```latex
Before the environments
\begin{one}
  first block, first line
  first block, second line
  first block, third line
\end{one}
\begin{two}
  body \end{two}
\end{one}
```

We can demonstrate interaction with the `-m` switch (see Section 6 on page 81); in particular, if we use Listing 457 on page 117, Listing 441 on page 115 and Listing 442 on page 115 and run

```bash
$ latexindent.pl --lines 6 myfile1.tex -o=+-mod1 -m -l env-mlb2, env-mlb7, env-mlb8 -o=+-mod1
```

then we receive the output in Listing 566.
LISTING 566: myfile1-mod1.tex

1 Before the environments
2 \begin{one}
3  first block, first line
4  first block, second line
5  first block, third line
6 \begin{two}
7  body
8 \end{two}
9 \end{one}
SECTION 9

Fine tuning

latexindent.pl operates by looking for the code blocks detailed in Table 2 on page 57. The fine tuning of the details of such code blocks is controlled by the fineTuning field, detailed in Listing 567. This field is for those that would like to peek under the bonnet/hood and make some fine tuning to latexindent.pl’s operating.

Warning!
Making changes to the fine tuning may have significant consequences for your indentation scheme, proceed with caution!

<table>
<thead>
<tr>
<th>Listing 567: fineTuning</th>
</tr>
</thead>
<tbody>
<tr>
<td>fineTuning:</td>
</tr>
<tr>
<td>environments:</td>
</tr>
<tr>
<td>name: [a-zA-Z-Z*0-9_]+</td>
</tr>
<tr>
<td>ifElseFi:</td>
</tr>
<tr>
<td>name: (?</td>
</tr>
<tr>
<td>commands:</td>
</tr>
<tr>
<td>name: [+a-zA-Z-Z*0-9_]+</td>
</tr>
<tr>
<td>items:</td>
</tr>
<tr>
<td>canBeFollowedBy: (?::[<em>]</em>?)</td>
</tr>
<tr>
<td>keyEqualsValuesBracesBrackets:</td>
</tr>
<tr>
<td>name: [a-zA-Z-Z*0-9_/.]+</td>
</tr>
<tr>
<td>follow: (?:(?!{)</td>
</tr>
<tr>
<td>namedGroupingBracesBrackets:</td>
</tr>
<tr>
<td>name: [0-9]+.a-zA-Z-Z*&lt;&gt;]+</td>
</tr>
</tbody>
</table>
| follow: \h\R\{|\(|\}
| UnNamedGroupingBracesBrackets: |
| follow: \{|\},\{|\}
| arguments:              |
| before: (?:#\d*;?,?/?)|\<.*?\> |
| between: _|\^-| |
| trailingComments:       |
| notPreceededBy: (?<!\) |
| afterComment: .*? |
| modifyLineBreaks:       |
| doubleBackSlash: \%(?! h\*;?,?/?)|\<.*?\> |
| comma: ',' |
| betterFullStop: |- |
| (?x) | # ignore spaces in the below |
| (?.  |
| \.) | # .) |
| (!\h* a-z)) | # not followed by a-z |
| ) | |
| | # OR |
| (?! | # |
| (?<! | # not preceded by |
| : | # |
| (?: | # |
| ([eE]\. [gG]) | # e.g OR E.g OR e.G OR E.G |
| | |

[git] • main @ 9220bdf • 2024-03-28 • V3.23.8
The fields given in Listing 567 are all regular expressions. This manual is not intended to be a tutorial on regular expressions; you might like to read, for example, [34] for a detailed covering of the topic.

We make the following comments with reference to Listing 567:

1. the environments:name field details that the name of an environment can contain:
   (a) a-z lower case letters
   (b) A-Z upper case letters
   (c) @ the @ 'letter'
   (d) \* stars
   (e) 0-9 numbers
   (f) _ underscores
   (g) \ backslashes
   The + at the end means at least one of the above characters.

2. the ifElseFi:name field:
   (a) @ means that it can possibly begin with @
   (b) followed by if
   (c) followed by 0 or more characters from a-z, A-Z and @
   (d) the ? the end means non-greedy, which means 'stop the match as soon as possible'

3. the keyEqualsValuesBracesBrackets contains some interesting syntax:
   (a) | means 'or'
   (b) (?:(?!\:\\)\{) the (?:...) uses a non-capturing group – you don't necessarily need to worry about what this means, but just know that for the fineTuning feature you should only ever use non-capturing groups, and not capturing groups, which are simply (...)
   (c) (?<!\:\\)\{ means a { but it can not be immediately preceded by a \n
4. in the arguments:before field
   (a) \d\h* means a digit (i.e. a number), followed by 0 or more horizontal spaces
   (b) ;?:,? means possibly a semi-colon, and possibly a comma
   (c) \<.*?\> is designed for 'beamer'-type commands; the .*? means anything in between <...>
5. the modifyLineBreaks field refers to fine tuning settings detailed in Section 6 on page 81. In particular:
   (a) betterFullStop is in relation to the one sentence per line routine, detailed in Section 6.2 on page 98
   (b) doubleBack Slash is in relation to the DBSStartsOnOwnLine and DBSFinishesWithLineBreak polyswitches surrounding double backslashes, see Section 6.3.2 on page 121
   (c) comma is in relation to the CommaStartsOnOwnLine and CommaFinishesWithLineBreak polyswitches surrounding commas in optional and mandatory arguments; see Table 3 on page 126

It is not obvious from Listing 567, but each of the follow, before and between fields allow trailing comments, line breaks, and horizontal spaces between each character.

**Warning!**

For the fineTuning feature you should only ever use non-capturing groups, such as (?::...) and not capturing groups, which are (...)

**example 164**

As a demonstration, consider the file given in Listing 568, together with its default output using the command

```
$ latexindent
```

is given in Listing 569.

**Listing 568: finetuning1.tex**

```
\mycommand{
  \rule{G -> +H[-G]CL}
  \rule{H -> -G[+H]CL}
  \rule{g -> +h[-g]cL}
  \rule{h -> -g[+h]cL}
}
```

**Listing 569: finetuning1.tex default**

```
\mycommand{
  \rule{G -> +H[-G]CL}
  \rule{H -> -G[+H]CL}
  \rule{g -> +h[-g]cL}
  \rule{h -> -g[+h]cL}
}
```

It's clear from Listing 569 that the indentation scheme has not worked as expected. We can **fine tune** the indentation scheme by employing the settings given in Listing 571 and running the command

```
$ latexindent
```

and the associated (desired) output is given in Listing 570.

**Listing 570: finetuning1.tex using Listing 571**

```
\mycommand{
  \rule{G -> +H[-G]CL}
  \rule{H -> -G[+H]CL}
  \rule{g -> +h[-g]cL}
  \rule{h -> -g[+h]cL}
}
```

**Listing 571: finetuning1.yaml**

```
fineTuning:
  arguments:
  between: '_|\^|\*|\->|\-|\+|h|H|g|G'
```

**example 165**

Let's have another demonstration; consider the file given in Listing 572, together with its default output using the command

```
$ latexindent
```
It's clear from Listing 573 that the indentation scheme has not worked as expected. We can fine tune the indentation scheme by employing the settings given in Listing 575 and running the command:

```bash
cmh:~$ latexindent.pl finetuning2.tex -l=fine-tuning2.yaml
```

and the associated (desired) output is given in Listing 574.

In particular, note that the settings in Listing 575 specify that NamedGroupingBracesBrackets and UnNamedGroupingBracesBrackets can follow " and that we allow --- between arguments.

**example 166**

You can tweak the fineTuning using the \texttt{-y} switch, but to be sure to use quotes appropriately. For example, starting with the code in Listing 576 and running the following command:

```bash
cmh:~$ latexindent.pl -m 
-y='modifyLineBreaks:oneSentencePerLine:manipulateSentences:;
 modifyLineBreaks:oneSentencePerLine:sentencesBeginWith:a-z;
 fineTuning:modifyLineBreaks:betterFullStop:
 "(?:\.|\?([a-z]))|(?:(?<!\(?:(?:e\.g)|(?:i\.e)|(?:etc)))\.|(?![a-z][A-Z])"
 issue-243.tex -o=+-mod1
```

gives the output shown in Listing 577.

**example 167**

We can tweak the fineTuning for how trailing comments are classified. For motivation, let's
consider the code given in Listing 578

\begin{lstlisting}[language=TeX]
\texttt{Listing 578: finetuning4.tex}
\end{lstlisting}

We will compare the settings given in Listings 579 and 580.

\begin{lstlisting}[language= YAML,frame=shadowbox]
\texttt{Listing 579: href1.yaml}
modifyLineBreaks:
\texttt{textWrapOptions:}
columns: -1
blocksEndBefore:
\texttt{verbatim: 0}
blocksFollow:
\texttt{verbatim: 0}
removeTrailingWhitespace:
\texttt{beforeProcessing: 1}
\end{lstlisting}

\begin{lstlisting}[language= YAML,frame=shadowbox]
\texttt{Listing 580: href2.yaml}
fineTuning:
\texttt{trailingComments:}
\texttt{notPrecededBy:}
\texttt{'(?:(?<!Handbook)(?!for)(?!Spoken))'}
modifyLineBreaks:
\texttt{textWrapOptions:}
columns: -1
blocksEndBefore:
\texttt{verbatim: 0}
blocksFollow:
\texttt{verbatim: 0}
removeTrailingWhitespace:
\texttt{beforeProcessing: 1}
\end{lstlisting}

Upon running the following commands

\begin{verbatim}
$ latexindent.pl -m finetuning4.tex -o=+-mod1 -l=href1
$ latexindent.pl -m finetuning4.tex -o=+-mod2 -l=href2
\end{verbatim}

we receive the respective output in Listings 581 and 582.

\begin{lstlisting}[language=TeX,frame=shadowbox]
\texttt{Listing 581: finetuning4.tex using Listing 579}
\end{lstlisting}

\begin{lstlisting}[language=TeX,frame=shadowbox]
\texttt{Listing 582: finetuning4.tex using Listing 580}
\end{lstlisting}

We note that in:

- Listing 581 the trailing comments are assumed to be everything following the first comment symbol, which has meant that everything following it has been moved to the end of the line; this is undesirable, clearly!
- Listing 582 has fine-tuned the trailing comment matching, and says that `%` cannot be immediately preceded by the words ‘Handbook’, ‘for’ or ‘Spoken’, which means that none of the `%` symbols have been treated as trailing comments, and the output is desirable.

\begin{example}
Another approach to this situation, which does not use fineTuning, is to use noIndentBlock which we discussed in Listing 44 on page 30; using the settings in Listing 583 and running the command

\begin{verbatim}
$ latexindent.pl -m finetuning4.tex -o=+-mod3 -l=href3
\end{verbatim}

then we receive the same output given in Listing 582.

\end{example}
With reference to the body field in Listing 583, we note that the body field can be interpreted as: the fewest number of zero or more characters that are not right braces. This is an example of character class.

**example 169**

We can use the fineTuning field to assist in the formatting of bibliography files.

Starting with the file in Listing 584 and running the command

```
$ latexindent .pl bib1.bib -o=+-mod1
```

gives the output in Listing 585.

```
@online{paulo,
title = "arararule,indent.yaml",
author = "PauloCereda",
date = {2013-05-23},
urldate = {2021-03-19},
keywords = {contributor},}
```

Let's assume that we would like to format the output so as to align the = symbols. Using the settings in Listing 587 and running the command

```
$ latexindent .pl bib1.bib -l bibsettings1.yaml -o=+-mod2
```

gives the output in Listing 586.

```
@online{paulo,
title = "arararule,indent.yaml",
author = "PauloCereda",
date = {2013-05-23},
urldate = {2021-03-19},
keywords = {contributor},}
```

Some notes about Listing 587:
• we have populated the lookForAlignDelims field with the online command, and have used the delimiterRegEx, discussed in Section 5.5.4 on page 44;

• we have tweaked the keyEqualsValuesBracesBrackets code block so that it will not be found following a comma; this means that, in contrast to the default behaviour, the lines such as date={2013-05-23}, will not be treated as key-equals-value braces;

• the adjustment to keyEqualsValuesBracesBrackets necessitates the associated change to the UnNamedGroupingBracesBrackets field so that they will be searched for following = symbols.

example 170

We can build upon Listing 587 for slightly more complicated bibliography files. 

Starting with the file in Listing 588 and running the command

```
cmh:~$ latexindent .pl bib2.bib -l bibsettings1.yaml -o=+-mod1
```

gives the output in Listing 589.

<table>
<thead>
<tr>
<th>LISTING 588: bib2.bib</th>
</tr>
</thead>
<tbody>
<tr>
<td>@online{cmh:videodemo,</td>
</tr>
<tr>
<td>title=“Videodemonstrationofpl.latexindentonyoutube”,</td>
</tr>
<tr>
<td>url=“<a href="https://www.youtube.com/watch?v=wo38aaH2F4E&amp;spfreload=10%E2%80%9D">https://www.youtube.com/watch?v=wo38aaH2F4E&amp;spfreload=10”</a>,</td>
</tr>
<tr>
<td>urldate={2017-02-21},</td>
</tr>
<tr>
<td>}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LISTING 589: bib2-mod1.bib</th>
</tr>
</thead>
<tbody>
<tr>
<td>@online{cmh:videodemo,</td>
</tr>
<tr>
<td>title = “Videodemonstrationofpl.latexindentonyoutube”,</td>
</tr>
<tr>
<td>url = &quot;<a href="https://www.youtube.com/watch?v=wo38aaH2F4E&amp;spfreload=10">https://www.youtube.com/watch?v=wo38aaH2F4E&amp;spfreload=10</a>&quot;,</td>
</tr>
<tr>
<td>urldate = {2017-02-21},</td>
</tr>
<tr>
<td>}</td>
</tr>
</tbody>
</table>

The output in Listing 589 is not ideal, as the = symbol within the url field has been incorrectly used as an alignment delimiter.

We address this by tweaking the delimiterRegEx field in Listing 590.

<table>
<thead>
<tr>
<th>LISTING 590: bibsettings2.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>lookForAlignDelims:</td>
</tr>
<tr>
<td>online:</td>
</tr>
<tr>
<td>delimiterRegEx: ‘(?&lt;!v)(?!spfreload)(=)’</td>
</tr>
</tbody>
</table>

Upon running the command

```
cmh:~$ latexindent .pl bib2.bib -l bibsettings1.yaml,bibsettings2.yaml -o=+-mod2
```

we receive the desired output in Listing 591.

<table>
<thead>
<tr>
<th>LISTING 591: bib2-mod2.bib</th>
</tr>
</thead>
<tbody>
<tr>
<td>@online{cmh:videodemo,</td>
</tr>
<tr>
<td>title = &quot;Videodemonstrationofpl.latexindentonyoutube&quot;,</td>
</tr>
<tr>
<td>url = &quot;<a href="https://www.youtube.com/watch?v=wo38aaH2F4E&amp;spfreload=10">https://www.youtube.com/watch?v=wo38aaH2F4E&amp;spfreload=10</a>&quot;,</td>
</tr>
<tr>
<td>urldate = {2017-02-21},</td>
</tr>
<tr>
<td>}</td>
</tr>
</tbody>
</table>
With reference to Listing 590 we note that the delimiterRegEx has been adjusted so that = symbols are used as the delimiter, but only when they are *not preceded* by either v or spf_reload.

**example 171**

We can use the fineTuning settings to tweak how latexindent.pl finds trailing comments.

We begin with the file in Listing 592:

```
\chapter{chapter text} % 123
chapter text
\section{section text} % 456
section text
% end
% end
```

Using the settings in Listing 594 and running the command

```
$ latexindent.pl finetuning5.tex -l=fine-tuning3.yaml
```

gives the output in Listing 593:

```
\chapter{chapter text} % 123
chapter text
\section{section text} % 456
section text
% end
% end
```

The settings in Listing 594 detail that trailing comments can *not* be followed by a single space, and then the text ‘end’. This means that the specialBeginEnd routine will be able to find the pattern `% end` as the end part. The trailing comments 123 and 456 are still treated as trailing comments.

**example 172**

We can use the fineTuning settings to tweak how latexindent.pl finds environments.

We begin with the file in Listing 595:

```
\begin{myenv}\label{mylabel}The body of my environment...\end{myenv}
```

Using the settings in Listing 597 and running the command

```
$ latexindent.pl finetuning6.tex -m -l=fine-tuning4.yaml
```

gives the output in Listing 596.
By using the settings in Listing 597 it means that the default poly-switch location of `BodyStartsOnOwnLine` for environments (denoted ♥ in Table 3) has been overwritten so that it is after the `label` command.

Referencing Listing 597, unless both `begin` and `end` are specified, then the default value of `name` will be used.
SECTION 10

Conclusions and known limitations

There are a number of known limitations of the script, and almost certainly quite a few that are unknown! The known issues include:

**multicolumn alignment** when working with code blocks in which multicolumn commands overlap, the algorithm can fail; see Listing 72 on page 37.

**textWrap after** when operating with indentRules (see Section 5.8 on page 56) may not always cooperate with one another; if you have a specific example that does not work, please report it to [35].

**efficiency** particularly when the \texttt{-m} switch is active, as this adds many checks and processes. The current implementation relies upon finding and storing every code block (see the discussion on page 130); I hope that, in a future version, only nested code blocks will need to be stored in the ‘packing’ phase, and that this will improve the efficiency of the script.

You can run \texttt{latexindent} on any file; if you don't specify an extension, then the extensions that you specify in fileExtensionPreference (see Listing 36 on page 27) will be consulted. If you find a case in which the script struggles, please feel free to report it at [35], and in the meantime, consider using a noIndentBlock (see page 30).

I hope that this script is useful to some; if you find an example where the script does not behave as you think it should, the best way to contact me is to report an issue on [35]; otherwise, feel free to find me on the \url{http://tex.stackexchange.com/users/6621/cmhughes}. 
SECTION 11

References

11.1 perl-related links

[33] Data::Dumper module. URL: https://perldoc.perl.org/Data::Dumper (visited on 06/18/2021).
[40] Log4perl Perl module. URL: http://search.cpan.org/~mschilli/Log-Log4perl-1.49/lib/Log/Log4perl.pm (visited on 09/24/2017).
[42] perldoc Encode::Supported. URL: https://perldoc.perl.org/Encode::Supported (visited on 05/06/2021).

11.2 conda-related links


11.3 VScode-related links


11.4 Other links

[38] How to use latexindent on Windows? URL: https://tex.stackexchange.com/questions/577250/how-to-use-latexindent-on-windows (visited on 01/08/2022).
[39] latexindent.pl ghcr (GitHub Container Repository) location. URL: https://github.com/cmhughes?tab=packages (visited on 06/12/2022).
11.5 Contributors (in chronological order)


[48] Video demonstration of latexindent.pl on youtube. URL: https://www.youtube.com/watch?v=wo38aH2F4E8&type=txt (visited on 02/21/2017).


Required Perl modules

If you intend to use `latexindent.pl` and not one of the supplied standalone executable files (`latexindent.exe` is available for Windows users without Perl, see Section 3.1.2), then you will need a few standard Perl modules.

If you can run the minimum code in Listing 598 as in

```bash
$ perl helloworld.pl
```

then you will be able to run `latexindent.pl`, otherwise you may need to install the missing modules; see appendices A.1 and A.2.

**Listing 598: helloworld.pl**

```perl
#!/usr/bin/perl

use strict; # |
use warnings; # |
use Encode; # |
use Getopt::Long; # |
use Data::Dumper; # generally part
use List::Util qw(max); # these modules are
use PerlIO::encoding; # of a default perl distribution
use open ':std', ':encoding(UTF-8)'; # |
use Text::Wrap; # |
use Text::Tabs; # |
use FindBin; # |
use File::Copy; # |
use File::Basename; # |
use File::Path; # |
use File::HomeDir; # <-- typically requires install via cpanm
use YAML::Tiny; # <-- typically requires install via cpanm

print "hello\nworld";
exit;
```

A.1 Module installer script

`latexindent.pl` ships with a helper script that will install any missing perl modules on your system; if you run

```bash
$ perl latexindent-module-installer.pl
```

or

```bash
C:\Users\cmh>perl latexindent-module-installer.pl
```

then, once you have answered Y, the appropriate modules will be installed onto your distribution.
A.2 Manually installing modules

Manually installing the modules given in Listing 598 will vary depending on your operating system and Perl distribution.

A.2.1 Linux

A.2.1.1 perlbrew

Linux users may be interested in exploring Perlbrew [41]; an example installation would be:

```
cmh:~$ sudo apt-get install perlbrew
cmh:~$ perlbrew init
cmh:~$ perlbrew install perl-5.34.0
```

```
cmh:~$ perlbrew switch perl-5.34.0
```

```
cmh:~$ sudo apt-get install curl
```

```
cmh:~$ curl -L http://cpanmin.us | perl - App::cpanminus
```

```
cmh:~$ cpanm YAML::Tiny
```

```
cmh:~$ cpanm File::HomeDir
```

A.2.1.2 Ubuntu/Debian

For other distributions, the Ubuntu/Debian approach may work as follows

```
cmh:~$ sudo apt install perl
```

```
cmh:~$ sudo cpan -i App::cpanminus
```

```
cmh:~$ sudo cpanm YAML::Tiny
```

```
cmh:~$ sudo cpanm File::HomeDir
```

or else by running, for example,

```
cmh:~$ sudo perl -MCPAN -e 'install "File::HomeDir"'
```

A.2.1.3 Ubuntu: using the texlive from apt-get

Ubuntu users that install texlive using apt-get as in the following

```
cmh:~$ sudo apt install texlive
```

```
cmh:~$ sudo apt install texlive-latex-recommended
```

may need the following additional command to work with `latexindent.pl`

```
cmh:~$ sudo apt install texlive-extra-utils
```

A.2.1.4 Ubuntu: users without perl

`latexindent-linux` is a standalone executable for Ubuntu Linux (and therefore does not require a Perl distribution) and caches copies of the Perl modules onto your system. It is available from [35].

```

A.2.1.5 Arch-based distributions

First install the dependencies

```
cmh:~$ sudo pacman -S perl cpanminus
```

In addition, install `perl-file-homedir` from AUR, using your AUR helper of choice,
A.2.1.6 Alpine

If you are using Alpine, some Perl modules are not build-compatible with Alpine, but replacements are available through apk. For example, you might use the commands given in Listing 599; thanks to [12] for providing these details.

```bash
# Listing 599: alpine-install.sh

# Installing perl
apk --no-cache add miniperl perl-utils

# Installing incompatible latexindent perl dependencies via apk
apk --no-cache add perl-log-dispatch perl-namespace-autoclean perl-specio perl-unicode-linebreak

# Installing remaining latexindent perl dependencies via cpan
apk --no-cache add curl wget make
ls /usr/share/texmf-dist/scripts/latexindent
cd /usr/local/bin && curl -L https://cpanmin.us/ -o cpanm &&
    chmod +x cpanm
cpanm -n App::cpanminus
cpanm -n File::HomeDir
cpanm -n Params::ValidationCompiler
cpanm -n YAML::Tiny
```

Users of NixOS might like to see https://github.com/cmhughes/latexindent.pl/issues/222 for tips.

A.2.2 Mac

Users of the Macintosh operating system might like to explore the following commands, for example:

```bash
# cmh:~$ brew install perl
# cmh:~$ brew install cpanm
# cmh:~$
# cmh:~$ cpanm YAML::Tiny
# cmh:~$ cpanm File::HomeDir
```

Alternatively,

```bash
# cmh:~$ brew install latexindent
```

latexindent-macos is a standalone executable for macOS (and therefore does not require a Perl distribution) and caches copies of the Perl modules onto your system. It is available from [35].

A.2.3 Windows

Strawberry Perl users on Windows might use CPAN client. All of the modules are readily available on CPAN [31]. indent.log will contain details of the location of the Perl modules on your system.

latexindent.exe is a standalone executable for Windows (and therefore does not require a Perl distribution) and caches copies of the Perl modules onto your system; if you wish to see where they are cached, use the trace option, e.g.
A.3 The GCString switch

If you find that the `lookForAlignDelims` (as in Section 5.5) does not work correctly for your language, then you may wish to use the `Unicode::GCString` module.

This can be loaded by calling `latexindent.pl` with the `GCString` switch as in

```
cmh:~$ latexindent.pl --GCString myfile.tex
```

In this case, you will need to have the `Unicode::GCString` installed in your `perl` distribution by using, for example,

```
cmh:~$ cpanm Unicode::GCString
```

Note: this switch does *nothing* for `latexindent.exe` which loads the module by default. Users of `latexindent.exe` should not see any difference in behaviour whether they use this switch or not, as `latexindent.exe` loads the `Unicode::GCString` module.
SECTION B

Updating the path variable

latexindent.pl has a few scripts (available at [35]) that can update the path variables. Thank you to [6] for this feature. If you're on a Linux or Mac machine, then you'll want CMakeLists.txt from [35].

B.1 Add to path for Linux

To add latexindent.pl to the path for Linux, follow these steps:

1. download latexindent.pl and its associated modules, defaultSettings.yaml, to your chosen directory from [35];

2. within your directory, create a directory called path-helper-files and download CMakeLists.txt and cmake_uninstall.cmake.in from [35]/path-helper-files to this directory;

3. run

```
$ ls /usr/local/bin
```

to see what is currently in there;

4. run the following commands

```
$ sudo apt-get update
$ sudo apt-get install --no-install-recommends cmake make # or any other generator
$ mkdir build && cd build
$ cmake ../path-helper-files
$ sudo make install
```

5. run

```
$ ls /usr/local/bin
```

again to check that latexindent.pl, its modules and defaultSettings.yaml have been added.

To remove the files, run

```
$ sudo make uninstall
```

B.2 Add to path for Windows

To add latexindent.exe to the path for Windows, follow these steps:

1. download latexindent.exe, defaultSettings.yaml, add-to-path.bat from [35] to your chosen directory;

2. open a command prompt and run the following command to see what is currently in your %path% variable;
3. right click on add-to-path.bat and Run as administrator;
4. log out, and log back in;
5. open a command prompt and run

C:\Users\cmh>echo %path%

To check that the appropriate directory has been added to your %path%.

To remove the directory from your %path%, run remove-from-path.bat as administrator.
SECTION C

Batches of files

You can instruct `latexindent.pl` to operate on multiple files. For example, the following calls are all valid

```
cmh:~$ latexindent.pl myfile1.tex
cmh:~$ latexindent.pl myfile1.tex myfile2.tex
cmh:~$ latexindent.pl myfile*.tex
```

We note the following features of the script in relation to the switches detailed in Section 3.

C.1 location of indent.log
If the `-c` switch is *not* active, then `indent.log` goes to the directory of the final file called. If the `-c` switch is active, then `indent.log` goes to the specified directory.

C.2 interaction with `-w` switch
If the `-w` switch is active, as in

```
cmh:~$ latexindent.pl -w myfile*.tex
```

then files will be overwritten individually. Back-up files can be re-directed via the `-c` switch.

C.3 interaction with `-o` switch
If `latexindent.pl` is called using the `-o` switch as in

```
cmh:~$ latexindent.pl myfile*.tex -o=my-output-file.tex
```

and there are multiple files to operate upon, then the `-o` switch is ignored because there is only one output file specified.

More generally, if the `-o` switch does *not* have a `+` symbol at the beginning, then the `-o` switch will be ignored, and is turned it off.

For example

```
cmh:~$ latexindent.pl myfile*.tex -o=myfile
```

will work fine because each `.tex` file will output to `<basename>myfile.tex`

Similarly,

```
cmh:~$ latexindent.pl myfile*.tex -o=++
```

will work because the 'existence check/incrementation' routine will be applied.

C.4 interaction with lines switch
This behaves as expected by attempting to operate on the line numbers specified for each file. See the examples in Section 8.
C.5 interaction with check switches

The exit codes for `latexindent.pl` are given in Table 1 on page 22.

When operating on multiple files with the check switch active, as in

```
cmh:~$ latexindent.pl myfile*.tex --check
```

then

- exit code 0 means that the text from *none* of the files has been changed;
- exit code 1 means that the text from *at least one* of the files been file changed.

The interaction with `checkv` switch is as in the check switch, but with verbose output.

C.6 when a file does not exist

What happens if one of the files can not be operated upon?

- if at least one of the files does not exist and `latexindent.pl` has been called to act upon multiple files, then the exit code is 3; note that `latexindent.pl` will try to operate on each file that it is called upon, and will not exit with a fatal message in this case;
- if at least one of the files can not be read and `latexindent.pl` has been called to act upon multiple files, then the exit code is 4; note that `latexindent.pl` will try to operate on each file that it is called upon, and will not exit with a fatal message in this case;
- if `latexindent.pl` has been told to operate on multiple files, and some do not exist and some cannot be read, then the exit code will be either 3 or 4, depending upon which it scenario it encountered most recently.
latexindent.pl ships with latexindent-yaml-schema.json which might help you when constructing your YAML files.

D.1 VSCode demonstration

To use latexindent-yaml-schema.json with VSCode, you can use the following steps:

1. download latexindent-yaml-schema.json from the documentation folder of \cite{35}, save it in whichever directory you would like, noting it for reference;

2. following the instructions from \cite{36}, for example, you should install the VSCode YAML extension \cite{49};

3. set up your settings.json file using the directory you saved the file by adapting Listing 600; on my Ubuntu laptop this file lives at /home/cmhughes/.config/Code/User/settings.json.

\begin{verbatim}
LISTING 600: settings.json
{
  "yaml.schemas": {
    "/home/cmhughes/projects/latexindent/documentation/latexindent-yaml-schema.json": 
    "/home/cmhughes/projects/latexindent/defaultSettings.yaml"
  },
  "redhat.telemetry.enabled": true
}
\end{verbatim}

Alternatively, if you would prefer not to download the json file, you might be able to use an adapted version of Listing 601.

\begin{verbatim}
LISTING 601: settings-alt.json
{
  "yaml.schemas": {
    "/home/cmhughes/projects/latexindent/defaultSettings.yaml"
  }
}
\end{verbatim}

Finally, if your TeX distribution is up to date, then latexindent-yaml-schema.json should be in the documentation folder of your installation, so an adapted version of Listing 602 may work.

\begin{verbatim}
LISTING 602: settings-alt1.json
{
  "yaml.schemas": {
    "/usr/local/texlive/2021/texmf-dist/doc/support/latexindent/latexindent-yaml-schema.json": 
    "/home/cmhughes/projects/latexindent/defaultSettings.yaml"
  }
}
\end{verbatim}

If you have details of how to implement this schema in other editors, please feel encouraged to contribute to this documentation.
SECTION E

Using conda

If you use conda you'll only need

```
cmh:~$ conda install latexindent.pl -c conda-forge
```

This will install the executable and all its dependencies (including perl) in the activate environment. You don't even have to worry about defaultSettings.yml as it included too, you can thus skip appendices A and B.

You can get a conda installation for example from [30] or from [29].

E.1 Sample conda installation on Ubuntu

On Ubuntu I followed the 64-bit installation instructions at [37] and then I ran the following commands:

```
cmh:~$ conda create -n latexindent.pl
cmh:~$ conda activate latexindent.pl
cmh:~$ conda install latexindent.pl -c conda-forge
cmh:~$ conda info --envs
cmh:~$ conda list
cmh:~$ conda run latexindent.pl -vv
```

I found the details given at [44] to be helpful.
If you use docker you’ll only need

```
cmh:~$ docker pull ghcr.io/cmhughes/latexindent.pl
```

This will download the image packed `latexindent`’s executable and its all dependencies. Thank you to [19] for contributing this feature; see also [39]. For reference, `ghcr` stands for GitHub Container Repository.

### F1 Sample docker installation on Ubuntu

To pull the image and show `latexindent`'s help on Ubuntu:

```
# setup docker if not already installed
if ! command -v docker &> /dev/null; then
  sudo apt install docker.io -y
  sudo groupadd docker
  sudo gpasswd -a "$USER" docker
  sudo systemctl restart docker
  newgrp docker
fi

# download image and execute
docker pull ghcr.io/cmhughes/latexindent.pl
docker run ghcr.io/cmhughes/latexindent.pl -h
```

Once I have run the above, on subsequent logins I run

```
newgrp docker
docker run ghcr.io/cmhughes/latexindent.pl -h
```

### F2 How to format on Docker

When you use `latexindent` with the docker image, you have to mount target `tex` file like this:

```
cmh:~$ docker run -v /path/to/local/myfile.tex:/myfile.tex
ghcr.io/cmhughes/latexindent.pl -s -w myfile.tex
```
SECTION G

pre-commit

Users of .git may be interested in exploring the pre-commit tool [43], which is supported by latexindent.pl. Thank you to [20] for contributing this feature, and to [21] for their contribution to it.

To use the pre-commit tool, you will need to install pre-commit; sample instructions for Ubuntu are given in appendix G.1. Once installed, there are two ways to use pre-commit: using CPAN or using conda, detailed in appendix G.3 and appendix G.4 respectively.

G.1 Sample pre-commit installation on Ubuntu

On Ubuntu I ran the following command:

```bash
$ python3 -m pip install pre-commit
```

I then updated my path via .bashrc so that it includes the line in Listing 605.

```
export PATH=$PATH:/home/cmhughes/.local/bin
```

G.2 pre-commit defaults

The default values that are employed by pre-commit are shown in Listing 606.

```
- id: latexindent
  name: latexindent.pl
  description: Run latexindent.pl (get dependencies using CPAN)
  minimum_pre_commit_version: 2.1.0
  entry: latexindent.pl
  args: ["--overwriteIfDifferent", "--silent", "--local"]
  language: perl
  types: [tex]
- id: latexindent-conda
  name: latexindent.pl
  description: Run latexindent.pl (get dependencies using Conda)
  minimum_pre_commit_version: 2.1.0
  entry: latexindent.pl
  args: ["--overwriteIfDifferent", "--silent", "--local"]
  language: conda
  types: [tex]
- id: latexindent-docker
  name: latexindent.pl
  description: Run latexindent.pl (get dependencies using Docker)
  minimum_pre_commit_version: 2.1.0
  entry: ghcr.io/cmhughes/latexindent.pl
  language: docker_image
  types: [tex]
  args: ["--overwriteIfDifferent", "--silent", "--local"]
```
In particular, the decision has deliberately been made (in collaboration with [21]) to have the default to employ the following switches: overwriteIfDifferent, silent, local; this is detailed in the lines that specify args in Listing 606.

**Warning!**

Users of pre-commit will, by default, have the overwriteIfDifferent switch employed. It is assumed that such users have version control in place, and are intending to overwrite their files.

### G.3 pre-commit using CPAN

To use latexindent.pl with pre-commit, create the file `.pre-commit-config.yaml` given in Listing 607 in your git-repository.

**Listing 607: `.pre-commit-config.yaml` (cpan)**

```
- repo: https://github.com/cmhughes/latexindent.pl
  rev: V3.23.8
  hooks:
    - id: latexindent
      args: [-s]
```

Once created, you should then be able to run the following command:

```
cmh:~$ pre-commit run --all-files
```

A few notes about Listing 607:

- the settings given in Listing 607 instruct pre-commit to use CPAN to get dependencies;
- this requires pre-commit and perl to be installed on your system;
- the args lists selected command-line options; the settings in Listing 607 are equivalent to calling

```
cmh:~$ latexindent.pl -s myfile.tex
```

for each .tex file in your repository;

- to instruct latexindent.pl to overwrite the files in your repository, then you can update Listing 607 so that args: [-s, -w].

Naturally you can add options, or omit -s and -w, according to your preference.

### G.4 pre-commit using conda

You can also rely on conda (detailed in appendix E) instead of CPAN for all dependencies, including latexindent.pl itself.

**Listing 608: `.pre-commit-config.yaml` (conda)**

```
- repo: https://github.com/cmhughes/latexindent.pl
  rev: V3.23.8
  hooks:
    - id: latexindent-conda
      args: [-s]
```

Once created, you should then be able to run the following command:

```
cmh:~$ pre-commit run --all-files
```

A few notes about Listing 607:

- the settings given in Listing 607 instruct pre-commit to use Conda to get dependencies;
- this requires pre-commit to be installed on your system;
- the args lists selected command-line options; the settings in Listing 607 are equivalent to calling

```
cmh:~$ latexindent.pl -s myfile.tex
```

for each .tex file in your repository;

- to instruct latexindent.pl to overwrite the files in your repository, then you can update Listing 607 so that args: [-s, -w].

Naturally you can add options, or omit -s and -w, according to your preference.
• the settings given in Listing 608 instruct pre-commit to use conda to get dependencies;
• this requires pre-commit and conda to be installed on your system;
• the args lists selected command-line options; the settings in Listing 607 are equivalent to calling

```
cmh:~$ conda run latexindent.pl -s myfile.tex
```
for each .tex file in your repository;
• to instruct latexindent.pl to overwrite the files in your repository, then you can update Listing 607 so that args: [-s, -w].

### G.5 pre-commit using docker

You can also rely on docker (detailed in appendix F) instead of CPAN for all dependencies, including latexindent.pl itself.

<table>
<thead>
<tr>
<th>LISTING 609: .pre-commit-config.yaml (docker)</th>
</tr>
</thead>
<tbody>
<tr>
<td>repo: <a href="https://github.com/cmhughes/latexindent.pl">https://github.com/cmhughes/latexindent.pl</a></td>
</tr>
<tr>
<td>rev: V3.23.8</td>
</tr>
<tr>
<td>hooks:</td>
</tr>
<tr>
<td>- id: latexindent-docker</td>
</tr>
<tr>
<td>args: [-s]</td>
</tr>
</tbody>
</table>

Once created, you should then be able to run the following command:

```
cmh:~$ pre-commit run --all-files
```

A few notes about Listing 607:
• the settings given in Listing 609 instruct pre-commit to use docker to get dependencies;
• this requires pre-commit and docker to be installed on your system;
• the args lists selected command-line options; the settings in Listing 607 are equivalent to calling

```
cmh:~$ docker run -v /path/to/myfile.tex:/myfile.tex ghcr.io/cmhughes/latexindent.pl -s myfile.tex
```
for each .tex file in your repository;
• to instruct latexindent.pl to overwrite the files in your repository, then you can update Listing 607 so that args: [-s, -w].

### G.6 pre-commit example using -l, -m switches

Let’s consider a small example, with local latexindent.pl settings in .latexindent.yaml.

**example 173**

We use the local settings given in Listing 610.

<table>
<thead>
<tr>
<th>LISTING 610: .latexindent.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>onlyOneBackUp: 1</td>
</tr>
<tr>
<td>modifyLineBreaks:</td>
</tr>
<tr>
<td>oneSentencePerLine:</td>
</tr>
<tr>
<td>manipulateSentences: 1</td>
</tr>
</tbody>
</table>

and .pre-commit-config.yaml as in Listing 611:
G.6 pre-commit example using -l, -m switches

Listing 611: `.pre-commit-config.yaml` (demo)

```yaml
- repo: https://github.com/cmhughes/latexindent.pl
  rev: V3.23.8
  hooks:
    - id: latexindent
      args: [-l, -m, -s, -w]
```

Now running

```
cmh:~$ pre-commit run --all-files
```

is equivalent to running

```
cmh:~$ latexindent.pl -l -m -s -w myfile.tex
```

for each `.tex` file in your repository.

A few notes about Listing 611:

- the `-l` option was added to use the local `.latexindent.yaml` (where it was specified to only create one back-up file, as `git` typically takes care of this when you use `pre-commit`);
- `-m` to modify line breaks; in addition to `-s` to suppress command-line output, and `-w` to format files in place.
SECTION H

indentconfig options

This section describes the possible locations for the main configuration file, discussed in Section 4. Thank you to [22] for this contribution.

The possible locations of indentconfig.yaml are read one after the other, and reading stops when a valid file is found in one of the paths.

Before stating the list, we give summarise in Table 5.

<table>
<thead>
<tr>
<th>environment variable</th>
<th>type</th>
<th>Linux</th>
<th>macOS</th>
<th>Windows</th>
</tr>
</thead>
<tbody>
<tr>
<td>LATEXINDENT_CONFIG</td>
<td>full path to file</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>XDG_CONFIG_HOME</td>
<td>directory path</td>
<td>✔</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>LOCALAPPDATA</td>
<td>directory path</td>
<td>✗</td>
<td>✗</td>
<td>✔</td>
</tr>
</tbody>
</table>

The following list shows the checked options and is sorted by their respective priority. It uses capitalized and with a dollar symbol prefixed names (e.g. $LATEXINDENT_CONFIG) to symbolize environment variables. In addition to that the variable name $homeDir is used to symbolize your home directory.

1. The value of the environment variable $LATEXINDENT_CONFIG is treated as highest priority source for the path to the configuration file.

2. The next options are dependent on your operating system:
   - Linux:
     (a) The file at $XDG_CONFIG_HOME/latexindent/indentconfig.yaml
     (b) The file at $homeDir/.config/latexindent/indentconfig.yaml
   - Windows:
     (a) The file at $LOCALAPPDATA\latexindent\indentconfig.yaml
     (b) The file at $homeDir\AppData\Local\latexindent\indentconfig.yaml
   - Mac:
     (a) The file at $homeDir/Library/Preferences/latexindent/indentconfig.yaml

3. The file at $homeDir/indentconfig.yaml

4. The file at $homeDir/.indentconfig.yaml

**H.1 Why to change the configuration location**

This is mostly a question about what you prefer, some like to put all their configuration files in their home directory (see $homeDir above), whilst some like to sort their configuration. And if you don’t care about it, you can just continue using the same defaults.
H.2 How to change the configuration location

This depends on your preferred location, if, for example, you would like to set a custom location, you would have to change the $LATEXINDENT_CONFIG environment variable.

Although the following example only covers $LATEXINDENT_CONFIG, the same process can be applied to $XDG_CONFIG_HOME or $LOCALAPPDATA because both are environment variables. You just have to change the path to your chosen configuration directory (e.g. $homeDir/.config or $homeDir/AppData\Local on Linux or Windows respectively).

H.2.1 Linux

To change $LATEXINDENT_CONFIG on Linux you can run the following command in a terminal after changing the path:

```bash
cmh:~$ echo 'export LATEXINDENT_CONFIG="/home/cmh/latexindent-config.yaml"' >> ~/.profile
```

Context: This command adds the given line to your .profile file (which is commonly stored in $HOME/.profile). All commands in this file are run after login, so the environment variable will be set after your next login.

You can check the value of $LATEXINDENT_CONFIG by typing

```bash
cmh:~$ echo $LATEXINDENT_CONFIG
```

Linux users interested in $XDG_CONFIG_HOME can explore variations of the following commands

```bash
cmh:~$ echo $XDG_CONFIG_HOME
```

```bash
cmh:~$ echo ${XDG_CONFIG_HOME:=$HOME/.config}
```

```bash
cmh:~$ mkdir /home/cmh/.config/latexindent
```

```bash
cmh:~$ touch /home/cmh/.config/indentconfig.yaml
```

H.2.2 Windows

To change $LATEXINDENT_CONFIG on Windows you can run the following command in powershell.exe after changing the path:

```bash
C:\Users\cmh>[Environment]:SetEnvironmentVariable
C:\Users\cmh>  ("LATEXINDENT_CONFIG", "\your\config\path", "User")
```

This sets the environment variable for every user session.

H.2.3 Mac

To change $LATEXINDENT_CONFIG on macOS you can run the following command in a terminal after changing the path:

```bash
cmh:~$ echo 'export LATEXINDENT_CONFIG="/your/config/path"' >> ~/.profile
```

Context: This command adds the line to your .profile file (which is commonly stored in $HOME/.profile). All commands in this file are run after login, so the environment variable will be set after your next login.
SECTION I

logFilePreferences

Listing 37 on page 28 describes the options for customising the information given to the log file, and we provide a few demonstrations here.

example 174

Let's say that we start with the code given in Listing 612, and the settings specified in Listing 613.

Listing 612: simple.tex
\begin{myenv}
body of myenv
\end{myenv}

Listing 613: logfile-prefs1.yaml

```
logFilePreferences:
  showDecorationStartCodeBlockTrace: "+++++
  showDecorationFinishCodeBlockTrace: "-----"
```

If we run the following command (noting that -t is active)

```
\$ latexindent.pl -t -l=logfile-prefs1.yaml simple.tex
```

then on inspection of indent.log we will find the snippet given in Listing 614.

Listing 614: indent.log

```
+++++
TRACE: environment found: myenv
No ancestors found for myenv
Storing settings for myenv
indentRulesGlobal specified (0) for environments, ...
Using defaultIndent for myenv
Putting linebreak after replacementText for myenv
looking for COMMANDS and key = {value}
TRACE: Searching for commands with optional and/or mandatory arguments AND key = {value}
looking for SPECIAL begin/end
TRACE: Searching myenv for special begin/end (see specialBeginEnd)
TRACE: Searching myenv for optional and mandatory arguments
... no arguments found
-----
```

Notice that the information given about myenv is ‘framed’ using ++++ and ----- respectively.
In relation to Section 4 on page 23, Windows users that encounter encoding issues with `indentconfig.yaml`, may wish to run the following command in either `cmd.exe` or `powershell.exe`:

```
C:\Users\cmh>chcp
```

They may receive the following result:

```
C:\Users\cmh>Active code page: 936
```

and can then use the settings given in Listing 615 within their `indentconfig.yaml`, where 936 is the result of the `chcp` command.

```
Listing 615: encoding demonstration for indentconfig.yaml

encoding: cp936
```

```
SECTION K

dos2unix linebreak adjustment

dos2unixlinebreaks: \{integer\}

If you use \texttt{latexindent.pl} on a dos-based Windows file on Linux then you may find that trailing horizontal space is not removed as you hope.

In such a case, you may wish to try setting \texttt{dos2unixlinebreaks} to 1 and employing, for example, the following command.

\begin{verbatim}
cmh:~$ latexindent.pl -y="dos2unixlinebreaks:1" myfile.tex
\end{verbatim}

See \cite{50} for further details.
SECTION L

Differences from Version 2.2 to 3.0

There are a few (small) changes to the interface when comparing Version 2.2 to Version 3.0. Explicitly, in previous versions you might have run, for example,

```bash
$ latexindent.pl -o myfile.tex outputfile.tex
```

whereas in Version 3.0 you would run any of the following, for example,

```bash
$ latexindent.pl -o=outputfile.tex myfile.tex
$ latexindent.pl myfile.tex -o=outputfile.tex
$ latexindent.pl myfile.tex -outputfile=outputfile.tex
$ latexindent.pl -outputfile outputfile.tex
```

noting that the output file is given next to the `-o` switch.

The fields given in Listing 616 are obsolete from Version 3.0 onwards.

<table>
<thead>
<tr>
<th>Listing 616: Obsolete YAML fields from Version 3.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>alwaysLookforSplitBrackets</td>
</tr>
<tr>
<td>alwaysLookforSplitBrackets</td>
</tr>
<tr>
<td>checkunmatched</td>
</tr>
<tr>
<td>checkunmatchedELSE</td>
</tr>
<tr>
<td>checkunmatchedbracket</td>
</tr>
<tr>
<td>constructIfElseFi</td>
</tr>
</tbody>
</table>

There is a slight difference when specifying indentation after headings; specifically, we now write `indentAfterThisHeading` instead of `indent`. See Listings 617 and 618

<table>
<thead>
<tr>
<th>Listing 617: indentAfterThisHeading in Version 2.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>indentAfterHeadings:</td>
</tr>
<tr>
<td>part:</td>
</tr>
<tr>
<td>indent: 0</td>
</tr>
<tr>
<td>level: 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 618: indentAfterThisHeading in Version 3.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>indentAfterHeadings:</td>
</tr>
<tr>
<td>part:</td>
</tr>
<tr>
<td>indentAfterThisHeading: 0</td>
</tr>
<tr>
<td>level: 1</td>
</tr>
</tbody>
</table>

To specify `noAdditionalIndent` for display-math environments in Version 2.2, you would write YAML as in Listing 619; as of Version 3.0, you would write YAML as in Listing 620 or, if you're using `-m` switch, Listing 621.
### Listing 619: noAdditionalIndent in Version 2.2

```latex
noAdditionalIndent:
\[ : 0
\] : 0
```

### Listing 620: noAdditionalIndent for displayMath in Version 3.0

```latex
specialBeginEnd:
displayMath:
begin: '\\['
end: '\\]'
lookForThis: 0
```

### Listing 621: noAdditionalIndent for displayMath in Version 3.0

```latex
noAdditionalIndent:
displayMath: 1
```

End
Listings

Listing 1: quick-start.tex .......................... 6
Listing 2: quick-start-default.tex ................... 6
Listing 3: quick-start-mod1.tex ..................... 7
Listing 4: quick-start1.yaml ....................... 7
Listing 5: quick-start-mod2.tex ..................... 7
Listing 6: quick-start2.yaml ....................... 7
Listing 7: quick-start-mod3.tex ..................... 8
Listing 8: quick-start3.yaml ....................... 8
Listing 9: quick-start-mod4.tex ..................... 8
Listing 10: quick-start4.yaml ...................... 8
Listing 11: quick-start-mod5.tex ................... 9
Listing 12: quick-start5.yaml ...................... 9
Listing 13: quick-start-mod6.tex ................... 9
Listing 14: quick-start6.yaml ...................... 9
Listing 15: quick-start-mod7.tex ................... 10
Listing 16: quick-start7.yaml ...................... 10
Listing 17: quick-start-mod8.tex ................... 10
Listing 18: quick-start8.yaml ..................... 10
Listing 19: quick-start-mod9.yaml ................. 11
Listing 20: quick-start9.yaml ..................... 11
Listing 21: Possible error messages ............... 11
Listing 22: demo.tex.tex .......................... 11
Listing 23: fileExtensionPreference ............... 11
Listing 24: modifyLineBreaks ...................... 12
Listing 25: replacements ........................... 12
Listing 26: filecontents1.tex ..................... 13
Listing 27: filecontents1.tex default output ...... 13
Listing 28: tikzset.tex ............................ 13
Listing 29: tikzset.tex default output ............ 13
Listing 30: pstricks.tex ............................ 13
Listing 31: pstricks.tex default output .......... 13
Listing 32: indentconfig.yaml (sample) .......... 23
Listing 33: mysettings.yaml (example) .......... 24
Listing 34: The encoding option for indentconfig.yaml .......... 24
Listing 36: fileExtensionPreference ............... 27
Listing 37: logFilePreferences .................... 28
Listing 38: verbatimEnvironments .................. 29
Listing 39: verbatimCommands ..................... 29
Listing 40: nameAsRegex1.yaml .................... 29
Listing 41: nameAsRegex2.yaml .................... 29
Listing 42: nameAsRegex3.yaml .................... 30
Listing 43: nameAsRegex4.yaml .................... 30
Listing 44: noIndentBlock ......................... 30
Listing 45: noIndentBlock.tex ..................... 30
Listing 46: noIndentBlock1.tex .................... 31
Listing 47: noindent1.yaml ....................... 31
Listing 48: noindent2.yaml ....................... 31
Listing 49: noindent3.yaml ....................... 31
Listing 50: noIndentBlock1.tex using Listing 47 or Listing 48 ....................... 31
Listing 51: noIndentBlock1.tex using Listing 49 ....................... 32
Listing 52: nameAsRegex5.yaml .................... 32
Listing 53: nameAsRegex6.yaml .................... 32
Listing 54: fileContentsEnvironments ............... 32
Listing 55: lookForPreamble ....................... 33
Listing 56: Motivating preambleCommandsBeforeEnvironments ....................... 33
Listing 57: removeTrailingWhitespace ............... 33
Listing 58: removeTrailingWhitespace (alt) .......... 33
Listing 60: tabular1.tex .......................... 34
Listing 61: tabular1.tex default output .......... 34
Listing 62: lookForAlignDelims (advanced) .......... 35
Listing 63: tabular2.tex .......................... 36
Listing 64: tabular2.yaml ......................... 36
Listing 65: tabular3.yaml ......................... 36
Listing 66: tabular4.yaml ......................... 36
Listing 67: tabular5.yaml ......................... 36
Listing 68: tabular6.yaml ......................... 36
Listing 69: tabular7.yaml ......................... 36
Listing 70: tabular8.yaml ......................... 36
Listing 71: tabular2.tex default output .......... 37
Listing 72: tabular2.tex using Listing 64 .......... 37
Listing 73: tabular2.tex using Listing 65 .......... 37
Listing 74: tabular2.tex using Listings 64 and 66 .......... 37
Listing 75: tabular2.tex using Listings 64 and 67 .......... 37
Listing 76: tabular2.tex using Listings 64 and 68 .......... 38
Listing 77: tabular2.tex using Listings 64 and 69 .......... 38
Listing 78: tabular2.tex using Listings 64 and 70 .......... 38
Listing 79: aligned1.tex ......................... 39
Listing 80: aligned1-default.tex .................. 39
Listing 81: sba1.yaml ............................. 39
Index

— B —
backup files
  cycle through, 25
  extension settings, 24
  maximum number of backup files, 25
  number of backup files, 25
  overwrite switch, -w, 12
  overwriteIfDifferent switch, -wd, 12
bibliography files, 146

— C —
capturing parenthesis (regex), 40
comments
  text wrap, 90, 105
conda, 11, 151, 160, 163
contributors, 151
cpan, 152, 163

— D —
delimiters, 117
  advanced settings, 31
  advanced settings of lookForAlignDelims, 30
  ampersand &, 31
  default settings of lookForAlignDelims, 31
  delimiter justification (left or right), 40
  delimiterRegEx, 40, 146
  dontMeasure feature, 38
  double backslash demonstration, 37
  lookForAlignDelims, 31
  poly-switches for double backslash, 115
  spacing demonstration, 33
  with specialBeginEnd and the -m switch, 117
  within specialBeginEnd blocks, 49
docker, 11, 161, 164

— E —
exit code, 18
  summary, 18

— G —
git, 163, 164

— I —
indention
  customising indentation per-code block, 53
  customising per-name, 53
  default, 15
  defaultIndent description, 30
  defaultIndent using -y switch, 15
  defaultIndent using YAML file, 20
  maximum indetation, 52
  no additional indent, 53
  no additional indent global, 53
  removing indentation per-code block, 53
  summary, 71

— J —
json
  schema for YAML files, 159
  VSCode, 159

— L —
latexindent-linux, 11, 152
latexindent-macos, 11, 153
latexindent.exe, 11
linebreaks
  summary of poly-switches, 117
  linux, 11, 152

— M —
macOS, 11, 153
MiKTeX, 11, 151
modifying linebreaks
  at the beginning of a code block, 106
  at the end of a code block, 109
  by using one sentence per line, 92
  surrounding double backslash, 115
  using poly-switches, 106

— P —
perl
  Unicode GCString module, 154
poly-switches
  adding comments and then line breaks: set to 2, 107, 109
  adding blank lines (again!): set to 4, 108
  adding blank lines: set to 3, 107
  adding blank lines (again!): set to 4, 110
  adding blank lines: set to 3, 110
  adding line breaks: set to 1, 106, 109
  blank lines, 113
  conflicting switches, 122
  conflicting partnering, 121
  conflicting switches, 123
  default values, 106
  definition, 106
  double backslash, 117
  environment global example, 106
  environment per-code block example, 106
  for double back slash (delimiters), 115
  for double backslash (delimiters), 117
  for double backslash (delimiters), 116, 118
off by default: set to 0, 106
removing line breaks: set to -1, 111
summary of all poly-switches, 119
values, 106
visualisation: ♠, ♥, ♦, ♣, 106

pre-commit, 151
conda, 163
cpan, 163
default, 162
docker, 164
warning, 163

— R —
regular expressions
capturing parenthesis, 40
captured class demonstration, 146
delimiter regex at \<= or \>, 41
delimiter regex at only \>, 41
don'tMeasure feature, cell, 39
don'tMeasure feature, row, 40
horizontal space \h, 103, 132
keyEqualsValuesBracesBrackets, 141
lowercase alph a-z, 39, 97
NamedGroupingBracesBrackets, 141
substitution field, arraycolsep, 128
substitution field, equation, 130
UnNamedGroupingBracesBrackets, 141
uppercase alph A-Z, 95, 103

regular expressions
a word about, 9
ampersand alignment, 146
ampersand alignment, 31
arguments, 141

at least one +, 131
at least one +, 49
at least one +, 128, 141–143
capturing parenthesis, 146
commands, 141
delimiter regex at #, 42
delimiter alignment for edge or node, 49
delimiter regex at # or \>, 42
delimiterRegEx, 31, 146
environments, 141
fine tuning, 141
horizontal space \h, 141
horizontal space \h, 49, 53, 131
ifElseFi, 141
lowercase alph a-z, 141
lowercase alph a-z, 40, 53, 92, 95, 103
modifyLineBreaks, 141
numeric 0-9, 53, 141
numeric 0-9, 49, 97, 103
replacement switch, -r, 127
text wrap
blocksFollow, 82, 83, 86
uppercase alph A-Z, 49
uppercase alph A-Z, 53, 92
uppercase alph A-Z, 141
using -y switch, 22

— S —
sentences
begin with, 95
begin with, 96
comments, 105
end with, 95, 97
follow, 95
indenting, 101
one sentence per line, 92
oneSentencePerLine, 92
removing sentence line breaks, 94
text wrap, 105
text wrapping, 101

specialBeginEnd
Algorithms example, 47
alignment at delimiter, 49
combined with lookForAlignDelims, 49
default settings, 44
delimiterRegEx, 49
delimiterRegEx tweaked, 49
double backslash poly-switch demonstration, 117
IFEsFi example, 46
indentRules example, 66
indentRulesGlobal, 71
introduction, 44
lookForAlignDelims, 117
middle, 46, 47
noAdditionalIndent, 66
noAdditionalIndentGlobal, 71
poly-switch summary, 119
searching for special before commands, 45
specifying as verbatim, 49
tikz example, 49
update to displaymath V3.0, 171

switches
–GCString, 18
–GCString demonstration, 154
–c, –cruft definition and details, 16
–d, –onlydefault definition and details, 16
–g, –logfile definition and details, 16
–h, –help definition and details, 12
–k, –check definition and details, 17
–kv, –checkv definition and details, 18
–l demonstration, 116, 118
–l demonstration, 22, 33, 39, 49, 52, 59, 61, 102, 128–130, 143, 144
–l in relation to other settings, 22
–l, –local definition and details, 14
–lines demonstration, 135
–lines demonstration, negation, 138, 139
–m demonstration, 108, 110, 112
–m demonstration, 92, 96, 98, 121, 122
–m demonstration, 77, 92, 94, 95, 97–99, 101–103, 107, 109, 113–118, 122–124, 130
–m, –modifylinebreaks definition and details, 16
–n, –lines definition and details, 18
–o demonstration, 37
–o demonstration, 42, 49, 92, 132, 171
–o, –output definition and details, 13
-r demonstration, 130  
-rr demonstration, 132  
-rr, –onlyreplacement definition and details, 17  
-rr demonstration, 132  
-rr demonstration, 129  
-s, –silent definition and details, 14  
-sl, –screenlog definition and details, 16  
-t, –trace definition and details, 14  
-tt, –ttrace definition and details, 14  
-v, –version definition and details, 12  
-vv, –vversion definition and details, 12  
-w, –overwrite definition and details, 12  
-wd, –overwriteIfDifferent definition and details, 12  
-rv demonstration, 132  
-rv, –replacementrespectverb definition and details, 17  
-rv, –replacementrespectverb switch, 17  
-rv, replacementrespectverb switch, 126  
-rv, replacementrespectverb switch, 17  
-y demonstration, 22, 37, 102  
-y, –yaml definition and details, 15

— T —
TeXLive, 11, 152, 153

text wrap
blocksFollow, 80
comments, 81
headings, 80
other, 82, 83, 86
comments, 90
text wrap
blocksBeginWith, 84
blocksEndBefore, 86
comments, 105
quick start, 79
setting columns to -1, 79

— V —
verbatim
commands, 26
comparison with -r and -rr switches, 132
environments, 26
in relation to oneSentencePerLine, 101
noIndentBlock, 27
poly-switch summary, 119
rv, replacementrespectverb switch, 126
rv, replacementrespectverb switch, 17
specialBeginEnd, 48
verbatimEnvironments demonstration (-l switch), 22
verbatimEnvironments demonstration (-y switch), 22
VSCode, 151, 159

— W —
warning
amalgamate field, 76
be sure to test before use, 2
capture groups, 143
capturing parenthesis for lookForAlignDelims, 40
changing huge (textwrap), 91
editing YAML files, 21