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; you can find a quick start guide in Section 1.4 on page 11.

Contents

1 Introduction 10
  1.1 Thanks ......................................................... 10
  1.2 License ......................................................... 10
  1.3 About this documentation .................................... 10
  1.4 Quick start .................................................. 11
  1.5 A word about regular expressions .......................... 12

2 Demonstration: before and after 13

3 How to use the script 14
  3.1 From the command line ....................................... 14
  3.2 From arara .................................................... 20

4 indentconfig.yaml, local settings and the -y switch 21
  4.1 indentconfig.yaml and .indentconfig.yaml .................... 21
  4.2 localSettings.yaml and friends ............................. 22
  4.3 The -y|yaml switch ........................................... 23
  4.4 Settings load order .......................................... 23

5 defaultSettings.yaml 25
  5.1 Backup and log file preferences ............................. 25

* and contributors! See Section 10.2 on page 133. For all communication, please visit [11].
6 The -m (modifylinebreaks) switch

6.1 Text Wrapping ............................. 73
  6.1.1 Text wrap quick start .................. 74
  6.1.2 textWrapOptions: modifying line breaks by text wrapping .......................... 74
  6.1.3 Text wrapping on a per-code-block basis ................................................. 77

6.2 removeParagraphLineBreaks: modifying line breaks for paragraphs .......................... 83

6.3 Combining removeParagraphLineBreaks and textWrapOptions .............................. 89
  6.3.1 text wrapping beforeFindingChildCodeBlocks ............................................ 90

6.4 Summary of text wrapping ................................................................. 92

6.5 oneSentencePerLine: modifying line breaks for sentences .............................. 92
  6.5.1 sentencesFollow ................................................................. 94
  6.5.2 sentencesBeginWith .............................................................. 95
  6.5.3 sentencesEndWith ............................................................... 96
  6.5.4 Features of the oneSentencePerLine routine ............................................ 98
  6.5.5 Text wrapping and indenting sentences .................................................... 99

6.6 Poly-switches .......................................................... 101
  6.6.1 Poly-switches for environments ......................................................... 101
    6.6.1.1 Adding line breaks: BeginStartsOnOwnLine and BodyStartsOnOwn-Line ................. 102
    6.6.1.2 Adding line breaks using EndStartsOnOwnLine and EndFinishesWithin-LineBreak ... 104
    6.6.1.3 poly-switches 1, 2, and 3 only add line breaks when necessary ..................... 105
    6.6.1.4 Removing line breaks (poly-switches set to −1) .................................... 106
    6.6.1.5 About trailing horizontal space ....................................................... 107
    6.6.1.6 poly-switch line break removal and blank lines .................................. 107
  6.6.2 Poly-switches for double back slash .................................................. 109
    6.6.2.1 Double back slash starts on own line ................................................. 109
    6.6.2.2 Double back slash finishes with line break ....................................... 110
    6.6.2.3 Double back slash poly-switches for specialBeginEnd ............................... 110
    6.6.2.4 Double back slash poly-switches for optional and mandatory arguments ......... 111
    6.6.2.5 Double back slash optional square brackets ....................................... 112
  6.6.3 Poly-switches for other code blocks .................................................. 112
  6.6.4 Partnering BodyStartsOnOwnLine with argument-based poly-switches ............... 114
6.6.5 Conflicting poly-switches: sequential code blocks .................... 115
6.6.6 Conflicting poly-switches: nested code blocks .......................... 116

7 The -r, -rv and -rr switches ............................... 118
  7.1 Introduction to replacements ........................................... 118
  7.2 The two types of replacements ......................................... 119
  7.3 Examples of replacements .............................................. 119

8 Fine tuning ........................................ 127

9 Conclusions and known limitations .................................. 132

10 References .................................................. 133
  10.1 External links ...................................................... 133
  10.2 Contributors .................................................... 133

A Required Perl modules ...................................... 135
  A.1 Module installer script .............................................. 135
  A.2 Manually installed modules ......................................... 135
    A.2.1 Linux .............................................................. 136
    A.2.2 Mac ............................................................... 136
    A.2.3 Windows .......................................................... 137

B Updating the path variable ...................................... 138
  B.1 Add to path for Linux .............................................. 138
  B.2 Add to path for Windows ........................................... 138

C logFilePreferences ........................................ 140

D Encoding indentconfig.yaml ...................................... 141

E dos2unix linebreak adjustment .................................. 142

F Differences from Version 2.2 to 3.0 .............................. 143

Index ..................................................................... 145

Listings

Listing 1: demo-tex.tex ........................................... 10
Listing 2: fileExtensionPreference ............................. 11
Listing 3: modifyLineBreaks ...................................... 11
Listing 4: replacements ............................................. 11
Listing 5: Possible error messages ............................. 11
Listing 6: filecontents1.tex ...................................... 13
Listing 7: filecontents1.tex default output ................... 13
Listing 8: tikzset.tex ................................................. 13
Listing 9: tikzset.tex default output ........................... 13
Listing 10: pstricks.tex ............................................ 13
Listing 11: pstricks.tex default output ....................... 13
Listing 12: The encoding option for indentconfig.yaml .... 22
Listing 14: The encoding option for indentconfig.yaml .... 22
Listing 16: fileExtensionPreference ........................... 25
Listing 17: logFilePreferences .................................... 26
Listing 18: verbatimEnvironments .............................. 27
Listing 19: verbatimCommands ..................................... 27
Listing 20: noIndentBlock .......................................... 27
Listing 21: noIndentBlock.tex ....................................... 28
Listing 22: noIndentBlock1.tex ..................................... 28
Listing 23: noindent1.yaml ......................................... 28
Listing 24: noindent2.yaml ............................ 28
Listing 25: noindent3.yaml ............................ 28
Listing 26: noIndentBlock1.tex using Listing 23 or Listing 24 ........................................ 28
Listing 27: noIndentBlock1.tex using Listing 25 ................................................................. 29
Listing 28: fileContentsEnvironments .......................... 29
Listing 29: lookForPreamble ........................................ 29
Listing 30: Motivating preambleCommandsBeforeEnvironments ........................................ 30
Listing 31: removeTrailingWhitespace ....................... 30
Listing 32: tabular1.tex ...................................... 31
Listing 33: tabular1.tex default output .......................... 31
Listing 34: lookForAlignDelims (advanced) .................... 31
Listing 35: tabular2.tex ...................................... 32
Listing 36: tabular2.yaml .................................... 32
Listing 37: tabular2.yaml default output .......................... 32
Listing 38: tabular2.tex default output .......................... 33
Listing 39: tabular3.yaml .................................... 32
Listing 40: tabular4.yaml .................................... 32
Listing 41: tabular5.yaml .................................... 32
Listing 42: tabular6.yaml .................................... 32
Listing 43: tabular7.yaml .................................... 32
Listing 44: tabular8.yaml .................................... 32
Listing 45: tabular2.tex default output .......................... 33
Listing 46: tabular2.tex using Listing 38 ....................... 33
Listing 47: tabular2.tex using Listing 39 ....................... 33
Listing 48: tabular2.tex using Listings 38 and 40 .......... 33
Listing 49: tabular2.tex using Listings 38 and 41 .......... 34
Listing 50: tabular2.tex using Listings 38 and 42 .......... 34
Listing 51: tabular2.tex using Listings 38 and 43 .......... 34
Listing 52: tabular2.tex using Listings 38 and 44 .......... 34
Listing 53: aligned1.tex ..................................... 35
Listing 54: aligned1-default.tex ............................... 35
Listing 55: sba1.yaml ....................................... 35
Listing 56: sba2.yaml ....................................... 35
Listing 57: sba3.yaml ....................................... 35
Listing 58: sba4.yaml ....................................... 35
Listing 59: aligned1-mod1.tex ................................. 36
Listing 60: sba5.yaml ....................................... 36
Listing 61: sba6.yaml ....................................... 36
Listing 62: aligned1-mod5.tex ................................. 36
Listing 63: aligned1.tex using Listing 64 ....................... 36
Listing 64: sba7.yaml ....................................... 36
Listing 65: tabular4.tex ..................................... 37
Listing 66: tabular4-default.tex ............................... 37
Listing 67: tabular4-FDBS.tex ................................ 37
Listing 68: matrix1.tex ...................................... 37
Listing 69: matrix1.tex default output .......................... 37
Listing 70: align-block.tex ................................... 37
Listing 71: align-block.tex default output .......................... 37
Listing 72: tabular-DM.tex ................................... 38
Listing 73: tabular-DM.tex default output .......................... 38
Listing 74: tabular-DM.tex using Listing 75 ....................... 38
Listing 75: dontMeasure1.yaml ............................... 38
Listing 76: tabular-DM.tex using Listing 77 or Listing 79 ........................................... 38
Listing 77: dontMeasure2.yaml ............................... 38
Listing 78: tabular-DM.tex using Listing 79 or Listing 79 ........................................... 38
Listing 79: dontMeasure3.yaml ............................... 39
Listing 80: dontMeasure4.yaml ............................... 39
Listing 81: tabular-DM.tex using Listing 82 ....................... 39
Listing 82: dontMeasure5.yaml ............................... 39
Listing 83: tabular-DM.tex using Listing 84 ....................... 39
Listing 84: dontMeasure6.yaml ............................... 39
Listing 85: tabbing.tex ..................................... 40
Listing 86: tabbing.tex default output .......................... 40
Listing 87: tabbing.tex using Listing 88 ....................... 40
Listing 88: delimiterRegEx1.yaml ............................ 40
Listing 89: tabbing.tex using Listing 90 ....................... 40
Listing 90: delimiterRegEx2.yaml ............................ 40
Listing 91: tabbing.tex using Listing 92 ....................... 40
Listing 92: delimiterRegEx3.yaml ............................ 41
Listing 93: tabbing1.tex ..................................... 41
Listing 94: tabbing1-mod4.tex ................................. 41
Listing 95: delimiterRegEx4.yaml ............................ 41
Listing 96: tabbing1-mod5.tex ................................. 41
Listing 97: delimiterRegEx5.yaml ............................ 41
Listing 98: indentAfterItems ................................... 42
Listing 99: items1.tex ....................................... 42
Listing 100: items1.tex default output .......................... 42
Listing 101: itemNames ....................................... 42
Listing 102: specialBeginEnd .................................. 42
Listing 103: special1.tex ..................................... 42
Listing 104: special1.tex default output .......................... 42
Listing 105: specialLR.tex .................................... 43
Listing 106: specialLeftRight.yaml ............................ 43
Listing 107: specialBeforeCommand.yaml .......................... 43
Listing 108: specialLR.tex using Listing 106 ....................... 43
Listing 109: specialLR.tex using Listings 106 and 107 ....................... 43
Listing 110: special2.tex ..................................... 44
Listing 111: middle.yaml ..................................... 44
Listing 112: special2.tex using Listing 111 ....................... 44
Listing 113: middle1.yaml ..................................... 44
Listing 114: special2.tex using Listing 113 ....................... 44
Listing 115: special-verbl.yaml ................................ 45
Listing 116: special3.tex and output using Listing 115 ....................... 45
Listing 117: special-align.tex ................................ 45
Listing 118: edge-node1.yaml ................................ 45
Listing 119: special-align.tex using Listing 118 ....................... 45
SECTION 1

Introduction

1.1 Thanks

I first created \texttt{latexindent.pl} to help me format chapter files in a big project. After I blogged about it on the \LaTeX\ stack exchange \cite{1} I received some positive feedback and follow-up feature requests. A big thank you to Harish Kumar \cite{15} who helped to develop and test the initial versions of the script.

The YAML-based interface of \texttt{latexindent.pl} was inspired by the wonderful \texttt{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 \texttt{latexindent.pl}, but the release of \texttt{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 10.2 on page 133 for their valued contributions, and thank you to those who report bugs and request features at \cite{11}.

1.2 License

\texttt{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 \texttt{latexindent.pl} has the option to overwrite your .\texttt{tex} files. It will always make at least one backup (you can choose how many it makes, see page 26) 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 9). You, the user, are responsible for ensuring that you maintain backups of your files before running \texttt{latexindent.pl} on them. I think it is important at this stage to restate an important part of the license here:

\textit{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 \cite{11} with a complete minimum working example as I would like to improve the code as much as possible.

\begin{warning}

Before you try the script on anything important (like your thesis), test it out on the sample files in the \texttt{test-case} directory \cite{11}.

\end{warning}

\textit{If you have used any version 2.* of \texttt{latexindent.pl}, there are a few changes to the interface; see appendix \texttt{F} on page 143 and the comments throughout this document for details.}

1.3 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 530. This may seem a lot, but I deem it necessary in presenting the various different options of \texttt{latexindent.pl} and the associated output that they are capable of producing.

The different listings are presented using different styles:

\begin{listing}

\texttt{demo-tex.tex} demonstration .\texttt{tex} file

This type of listing is a .\texttt{tex} file.
\end{listing}
1.4 Quick start

If you’d like to get started with `latexindent.pl` then simply type

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

from the command line. If you receive an error message such as that given in Listing 5, then you need to install the missing perl modules.

```
Can't locate File/HomeDir.pm in @INC (INC contains:...)
```

`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.

```
BEGIN failed -- compilation aborted at helloworld.pl line 10.
```
1.5 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 [10].

You might also like to see https://stackoverflow.com/questions/19590042/error-cant-locate-file-homedir-pm-in-inc, for example, as well as appendix A on page 135.
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 [27].

As you look at Listings 6 to 11, 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 6 to 11 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).

<table>
<thead>
<tr>
<th>Listing 6: filecontents1.tex</th>
<th>Listing 7: filecontents1.tex default output</th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{filecontents}{mybib.bib} @online{strawberryperl, title=&quot;Strawberry Perl&quot;, url=&quot;<a href="http://strawberryperl.com/%22%7D">http://strawberryperl.com/&quot;}</a> @online{cmhblog, title=&quot;A Perl script ... url=&quot;...} \end{filecontents}</td>
<td>\begin{filecontents}{mybib.bib} @online{strawberryperl, title=&quot;Strawberry Perl&quot;, url=&quot;<a href="http://strawberryperl.com/%22%7D">http://strawberryperl.com/&quot;}</a> @online{cmhblog, title=&quot;A Perl script ... url=&quot;...} \end{filecontents}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 8: tikzset.tex</th>
<th>Listing 9: tikzset.tex default output</th>
</tr>
</thead>
<tbody>
<tr>
<td>\tikzset{ shrink inner sep/.code={ \pgfkeysgetvalue\pgfkeysgetvalue... } }</td>
<td>\tikzset{ shrink inner sep/.code={ \pgfkeysgetvalue\pgfkeysgetvalue... } }</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 10: pstricks.tex</th>
<th>Listing 11: pstricks.tex default output</th>
</tr>
</thead>
<tbody>
<tr>
<td>\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}}</td>
<td>\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}}</td>
</tr>
</tbody>
</table>
SECTION 3

How to use the script

latexindent.pl ships as part of the Ti\TeX{}Live distribution for Linux and Mac users; latexindent.exe ships as part of the Ti\TeX{}Live and MiK\TeX{} distributions for Windows users. These files are also available from github[11] should you wish to use them without a Ti\TeX{} distribution; in this case, you may like to read appendix B on page 138 which details how the path variable can be updated.

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

There are two ways to use latexindent.pl: from the command line, and using arara; we discuss these in Section 3.1 and Section 3.2 respectively. We will discuss how to change the settings and behaviour of the script in Section 5 on page 25.

latexindent.pl ships with latexindent.exe for Windows users, so that you can use the script with or without a Perl distribution. If you plan to use latexindent.pl (i.e., the original Perl script) then you will need a few standard Perl modules – see appendix A on page 135 for details; in particular, note that a module installer helper script is shipped with latexindent.pl.

3.1 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

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

This will output only the version number to the terminal.

-h, --help

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

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

```
cmh:~$ 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.

-w, --overwrite
3.1 From the command line

```bash
cmh:~$ latexindent.pl -w myfile.tex
cmh:~$ latexindent.pl --overwrite myfile.tex
cmh:~$ 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.

```bash
-o=output.tex,-outputfile=output.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).

Note that using -o as above is equivalent to using

```bash
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:

```bash
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:

```bash
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,

```bash
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

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

¹Users of version 2.* should note the subtle change in syntax.
tells it to output to myfile0.tex, but if it exists then output to myfile1.tex and so on.

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

```
cmh:~$ 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.

There is no need to specify a file extension when using the ++ feature, but if you wish to, then you should include it after the ++ symbols, for example

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

See appendix F on page 143 for details of how the interface has changed from Version 2.2 to Version 3.0 for this flag.

- **-s, –silent**

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

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

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

- **-t, –trace**

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

cmh:~$ 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, –ttrace**

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

cmh:~$ 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,...]**

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

cmh:~$ latexindent.pl -l=first.yaml,second.yaml,third.yaml myfile.tex

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 22) in the current
3.1 From the command line

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 com-
mas) of a YAML file(s) that resides in the same directory as myfile.tex; you can use this op-
tion 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

```bash
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:

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

which translate, respectively, to

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

Note that the following is not allowed:

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

and

```bash
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

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

-y, --yaml=yaml settings

```bash
cmh:~$ latexindent.pl myfile.tex -y="defaultIndent:\u007b\u007du'\u007d"
cmh:~$ latexindent.pl myfile.tex -y="defaultIndent:\u007b\u007du',maximumIndentation:\u007d'\u007d"
cmh:~$ latexindent.pl myfile.tex -y="indentRules:\u007bone:\u007d\t\t\t\t\t"'
cmh:~$ latexindent.pl myfile.tex
-y="modifyLineBreaks:environments:EndStartsOnOwnLine:3' -m
```

```bash
cmh:~$ latexindent.pl myfile.tex
-y="modifyLineBreaks:environments:one:EndStartsOnOwnLine:3' -m
```
You can specify YAML settings from the command line using the \texttt{-y} or \texttt{--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 \texttt{;} to separate fields, which is demonstrated in Section 4.3 on page 23.

Any settings specified via this switch will be loaded after any specified using the \texttt{-l} switch. This is discussed further in Section 4.4 on page 23.

\texttt{-d, \texttt{--onlydefault}}

```
$ latexindent.pl \texttt{-d} myfile.tex
```

Only defaultSettings.yaml: you might like to read Section 5 before using this switch. By default, \texttt{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 \texttt{-d} switch; note that this will also tell the script to ignore localSettings.yaml even if it has been called with the \texttt{-l} switch; \texttt{latexindent.pl} will also ignore any settings specified from the \texttt{-y} switch.

\texttt{-c, \texttt{--cruft=<directory>}}

```
$ latexindent.pl \texttt{-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.

\texttt{-g, \texttt{--logfile=<name of log file>}}

```
$ latexindent.pl \texttt{-g=other.log} myfile.tex
$ latexindent.pl \texttt{--logfile other.log} myfile.tex
$ latexindent.pl myfile.tex \texttt{-g other.log}
```

By default, \texttt{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 \texttt{-g} switch as demonstrated above.

If \texttt{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

```
$ latexindent.pl \texttt{-g /dev/null} myfile.tex
```

\texttt{-sl, \texttt{--screenlog}}

```
$ latexindent.pl \texttt{-sl} myfile.tex
$ latexindent.pl \texttt{--screenlog} myfile.tex
```

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

\texttt{-m, \texttt{--modifylinebreaks}}

```
$ latexindent.pl \texttt{-m} myfile.tex
$ latexindent.pl \texttt{--modifylinebreaks} myfile.tex
```
3.1 From the command line

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 72.

`latexindent.pl` can also be called on a file without the file extension, for example

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

and in which case, you can specify the order in which extensions are searched for; see Listing 16 on page 25 for full details.

### STDIN

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

`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:

```bash
cmh:~$ cat myfile.tex | latexindent.pl -l=mysettings.yaml -
```

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

```bash
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 [4] for an update to this feature.

### -r, --replacement

```bash
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 118.

### -rv, --replacementrespectverb

```bash
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 118.

### -rr, --onlyreplacement
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 118.

### 3.2 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 [3].
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.

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 [9] 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\username. Listing 12 shows a sample indentconfig.yaml file.

```
# Paths to user settings for latexindent.pl
#
# Note that the settings will be read in the order you
# specify here- each successive settings file will overwrite
# the variables that you specify

paths:
- /home/cmhughes/Documents/yamlfiles/mysettings.yaml
- /home/cmhughes/folder/othersettings.yaml
- /some/other/folder/anynameyouwant.yaml
- C:\Users\chughes\Documents\mysettings.yaml
- C:\Users\chughes\Desktop\test spaces\more spaces.yaml
```

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 13 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.

---

[9] If 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.
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 latexindent.pl can not read your .yaml file it will tell you so in indent.log.

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 14.

If you’d prefer to name your localSettings.yaml file something different, (say, mysettings.yaml as in Listing 13) then you can call latexindent.pl using, for example,

```
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 13) then you can call latexindent.pl using, for example,
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 15 using the -y switch, then you could use the following command:

```
$ 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:

```
$ 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,

```
$ 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 6.5 on page 92) and the listings within Listing 335 on page 94, the following settings give the option to have sentences end with a semicolon

```
$ latexindent.pl --yaml="/quotesingle.ts1
 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.
SECTION 5

defaultSettings.yaml

latexindent.pl loads its settings from defaultSettings.yaml. The idea is to separate the behaviour 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

```
$c mh:~$ 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.

```
<table>
<thead>
<tr>
<th>LISTING 16: fileExtensionPreference</th>
</tr>
</thead>
<tbody>
<tr>
<td>44 fileExtensionPreference:</td>
</tr>
<tr>
<td>45 .tex: 1</td>
</tr>
<tr>
<td>46 .sty: 2</td>
</tr>
<tr>
<td>47 .cls: 3</td>
</tr>
<tr>
<td>48 .bib: 4</td>
</tr>
</tbody>
</table>
```

Calling latexindent.pl myfile with the (default) settings specified in Listing 16 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 \texttt{-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 \texttt{-w} to act upon myfile.tex, it will create successive back up files: myfile.bak1, myfile.bak2, etc.

---

4Throughout 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
cmh:∼$ copy myfile.bak3 to myfile.bak2
cmh:∼$ copy myfile.bak4 to myfile.bak3
```

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 17. If you load your own user settings (see Section 4 on page 21) 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 17: 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 16) 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 C on page 140.
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 17; see [7] and [6] for more information. These options will mostly be helpful for those calling latexindent.pl with the -tt option described in Section 3.1.

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 18.

| Listing 18: verbatimEnvironments |
|-----------------|-----------------|
| 106  | verbatimEnvironments: |
| 107  | \verbatim: 1 |
| 108  | lstlisting: 1 |
| 109  | minted: 1 |

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.

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 72).

With reference to Listing 19, 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.

noIndentBlock: (fields)

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 20.

<table>
<thead>
<tr>
<th>Listing 20: noIndentBlock</th>
</tr>
</thead>
<tbody>
<tr>
<td>119</td>
</tr>
<tr>
<td>120</td>
</tr>
<tr>
<td>121</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 21 for example.
5.2 Verbatim code blocks

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

The noIndentBlock fields can also be specified in terms of begin and end fields. We use the code in Listing 22 to demonstrate this feature.

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

The settings given in Listings 23 and 24 are equivalent:

```yaml
noIndentBlock:
  demo:
    begin: '\some\hbefore'
    body: '.*?'
    end: '\some\hafter\htext'
    lookForThis: 1
```

```yaml
noIndentBlock:
  demo:
    begin: '\some\hbefore'
    end: '\some\hafter\htext'
```

```yaml
noIndentBlock:
  demo:
    begin: '\some\hbefore'
    body: '.*?'
    end: '\some\hafter\htext'
    lookForThis: 0
```

Upon running the commands

```
cmh:~$ latexindent.pl -l noindent1.yaml noindent1
cmh:~$ latexindent.pl -l noindent2.yaml noindent1
```

then we receive the output given in Listing 26.

```latex
\begin{noindent}
\begin{verbatim}
\end{verbatim}
\end{noindent}
```

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 .*? which is written explicitly in Listing 23. In this context, we interpret .*? 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).
Using Listing 25 demonstrates setting lookForThis to 0 (off); running the command

```
$ latexindent.pl -l noindent3.yaml noindent1
```

gives the output in Listing 27.

```
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 518.

### 5.3 filecontents and preamble

**fileContentsEnvironments:** (field)

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 28. The behaviour of latexindent.pl on these environments is determined by their location (preamble or not), and the value indentPreamble, discussed next.

```
Listing 28: fileContentsEnvironments

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>125</td>
<td>fileContentsEnvironments:</td>
</tr>
<tr>
<td>126</td>
<td>filecontents: 1</td>
</tr>
<tr>
<td>127</td>
<td>filecontents*: 1</td>
</tr>
</tbody>
</table>
```

**indentPreamble:** 0|1

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 29, 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.

```
Listing 29: lookForPreamble

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>133</td>
<td>lookForPreamble:</td>
</tr>
<tr>
<td>134</td>
<td>.tex: 1</td>
</tr>
<tr>
<td>135</td>
<td>.sty: 0</td>
</tr>
<tr>
<td>136</td>
<td>.cls: 0</td>
</tr>
<tr>
<td>137</td>
<td>.bib: 0</td>
</tr>
</tbody>
</table>
```

**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 30.

```
... preheadhook={\begin{mdframed}[style=myframedstyle]},
postfoothook=\end{mdframed}, ...
```

### 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 \tab which means a tab; we will explore customisation beyond defaultIndent in Section 5.8 on page 48.

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 31; each of the fields can take the values 0 or 1. See Listings 418 to 420 on page 107 for before and after results. Thanks to [28] for providing this feature.

```
150 removeTrailingWhitespace:
151   beforeProcessing: 0
152   afterProcessing: 1
```

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 32.

### 5.5 Aligning at delimiters

**lookForAlignDelims:** *(fields)*

This contains a list of code blocks that are operated upon in a special way by `latexindent.pl` (see Listing 33). In fact, the fields in lookForAlignDelims can actually take two different forms: the *basic* version is shown in Listing 33 and the *advanced* version in Listing 36; we will discuss each in turn.

```
lookForAlignDelims:
  tabular: 1
  tabularx: 1
  longtable: 1
  array: 1
  matrix: 1
  ...
```
Specifying code blocks in this field instructs latexindent.pl to try and align each column by its alignment delimiters. It does have some limitations (discussed further in Section 9), but in many cases it will produce results such as those in Listings 34 and 35.

If you find that latexindent.pl does not perform satisfactorily on such environments then you can set the relevant key to 0, for example tabular: 0; alternatively, if you just want to ignore specific instances of the environment, you could wrap them in something from noIndentBlock (see Listing 20 on page 27).

If, for example, you wish to remove the alignment of the within a delimiter-aligned block, then the advanced form of lookForAlignDelims shown in Listing 36 is for you.

Note that you can use a mixture of the basic and advanced form: in Listing 36 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 33. 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 delimeters 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);
5.5 Aligning at delimiters

- justification: optionally specifies the justification of each cell as either left or right (default: left);
- alignFinalDoubleBackSlash optionally specifies if the final double back slash 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.

We will explore most of these features using the file `tabular2.tex` in Listing 37 (which contains a `\multicolumn` command), and the YAML files in Listings 38 to 44; we will explore `alignFinalDoubleBackSlash` in Listing 65; the `dontMeasure` feature will be described in Section 5.5.3, and `delimiterRegEx` in Section 5.5.4.

---

**LISTING 37: tabular2.tex**

```
\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 38: tabular2.yaml**

```
lookForAlignDelims:
  tabular:
    multiColumnGrouping: 1
```

**LISTING 39: tabular3.yaml**

```
lookForAlignDelims:
  tabular:
    alignRowsWithoutMaxDelims: 0
```

**LISTING 40: tabular4.yaml**

```
lookForAlignDelims:
  tabular:
    spacesBeforeAmpersand: 4
```

**LISTING 41: tabular5.yaml**

```
lookForAlignDelims:
  tabular:
    spacesAfterAmpersand: 4
```

**LISTING 42: tabular6.yaml**

```
lookForAlignDelims:
  tabular:
    alignDoubleBackSlash: 0
```

**LISTING 43: tabular7.yaml**

```
lookForAlignDelims:
  tabular:
    spacesBeforeDoubleBackSlash: 0
```

**LISTING 44: tabular8.yaml**

```
lookForAlignDelims:
  tabular:
    justification: "right"
```

On running the commands
we obtain the respective outputs given in Listings 45 to 52.

**Listing 45:** tabular2.tex default output

```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 46:** tabular2.tex using Listing 38

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

**Listing 47:** tabular2.tex using Listing 39

```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 48:** tabular2.tex using Listings 38 and 40

```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}
```
5.5 Aligning at delimiters

Listing 49: \texttt{tabular2.tex} using Listings 38 and 41

\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 50: \texttt{tabular2.tex} using Listings 38 and 42

\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 51: \texttt{tabular2.tex} using Listings 38 and 43

\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 52: \texttt{tabular2.tex} using Listings 38 and 44

\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 45 and 46 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 45 the columns have been aligned at the ampersand;
- in Listing 46 the \texttt{multicolumn} command has grouped the 2 columns beneath and above it, because \texttt{multiColumnGrouping} is set to 1 in Listing 38;
- in Listing 47 rows 3 and 6 have not been aligned at the ampersand, because \texttt{alignRowsWithoutMaxDelims} has been set to 0 in Listing 39; however, the \texttt{\\} have still been aligned;
- in Listing 48 the columns beneath and above the \texttt{multicolumn} commands have been grouped (because \texttt{multiColumnGrouping} is set to 1), and there are at least 4 spaces \texttt{before} each aligned ampersand because \texttt{spacesBeforeAmpersand} is set to 4;
- in Listing 49 the columns beneath and above the \texttt{multicolumn} commands have been grouped (because \texttt{multiColumnGrouping} is set to 1), and there are at least 4 spaces \texttt{after} each aligned ampersand because \texttt{spacesAfterAmpersand} is set to 4;
• in Listing 50 the `\` have not been aligned, because `alignDoubleBackSlash` is set to 0, otherwise the output is the same as Listing 46;

• in Listing 51 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 46;

• in Listing 52 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 38.

5.5.1 `lookForAlignDelims: spacesBeforeAmpersand`

The `spacesBeforeAmpersand` can be specified in a few different ways. The `basic` form is demonstrated in Listing 40, 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.

We demonstrate this feature in relation to Listing 53; upon running the following command

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

then we receive the default output given in Listing 54.

<table>
<thead>
<tr>
<th>Listing 53: aligned1.tex</th>
<th>Listing 54: 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 55 to 58 are all equivalent; we have used the not-yet discussed `noAdditionalIndent` field (see Section 5.8 on page 48) which will assist in the demonstration in what follows.

<table>
<thead>
<tr>
<th>Listing 55: sba1.yaml</th>
<th>Listing 56: 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: 1</td>
</tr>
<tr>
<td></td>
<td>spacesBeforeAmpersand:</td>
</tr>
<tr>
<td></td>
<td>default: 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 57: sba3.yaml</th>
<th>Listing 58: 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>default: 1</td>
</tr>
<tr>
<td></td>
<td>leadingBlankColumn: 1</td>
</tr>
</tbody>
</table>

Upon running the following commands

```bash
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 59; we note that there is one space before each ampersand.
5.5 Aligning at delimiters

LISTING 59: aligned1-mod1.tex
\begin{aligned}
& a & b, \\
& c & d.
\end{aligned}

We note in particular:

- Listing 55 demonstrates the basic form for lookForAlignDelims; in this case, the default values are specified as in Listing 36 on page 31;
- Listing 56 demonstrates the advanced form for lookForAlignDelims and specified spacesBeforeAmpersand. The default value is 1;
- Listing 57 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 57, and it will inherit the value from default;

- Listing 58 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.

We can customise the space before the ampersand in the leading blank column of Listing 59 by using either of Listings 60 and 61, which are equivalent.

<table>
<thead>
<tr>
<th>LISTING 60: sba5.yaml</th>
<th>LISTING 61: 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>leadingBlankColumn: 0</td>
<td>leadingBlankColumn: 0</td>
</tr>
<tr>
<td>default: 1</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 62. We note that the space before the ampersand in the leading blank column has been set to 0 by Listing 61.

We can demonstrated this feature further using the settings in Listing 64 which give the output in Listing 63.

<table>
<thead>
<tr>
<th>LISTING 62: aligned1-mod5.tex</th>
<th>LISTING 63: aligned1.tex using Listing 64</th>
<th>LISTING 64: sba7.yaml</th>
</tr>
</thead>
</table>
| \begin{aligned}
& a & b, \\
& c & d.
\end{aligned} | \begin{aligned}
& a & b, \\
& c & d.
\end{aligned} | noAdditionalIndent: |
|                               |                                           | aligned: 1            |
|                               |                                           | lookForAlignDelims:   |
|                               |                                           | aligned:              |
|                               |                                           | spacesBeforeAmpersand:|
|                               |                                           | leadingBlankColumn: 3 |
|                               |                                           | default: 0            |

5.5.2 lookForAlignDelims: alignFinalDoubleBackSlash

We explore the alignFinalDoubleBackSlash feature by using the file in Listing 65. Upon running the following commands
5.5 Aligning at delimiters

We note that in:

- Listing 66, by default, the first set of double back slashes in the first row of the \texttt{tabular} environment have been used for alignment;
- Listing 67, the final set of double back slashes in the first row have been used, because we specified \texttt{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 \texttt{specialBeginEnd} on page 42); for example, assuming that you have a command called \texttt{\textbackslash matrix} and that it is populated within \texttt{lookForAlignDelims} (which it is, by default), and that you run the command

\begin{verbatim}
cmh:~$ latexindent.pl matrix1.tex
\end{verbatim}

then the before-and-after results shown in Listings 68 and 69 are achievable by default.

If you have blocks of code that you wish to align at the \& character that are not wrapped in, for example, \texttt{\begin{tabular}}...\texttt{\end{tabular}}, then you can use the mark up illustrated in Listing 70; the default output is shown in Listing 71. Note that the \% must be next to each other, but that there can be any number of spaces (possibly none) between the \% and \texttt{\begin{tabular}}; note also that you may use any environment name that you have specified in \texttt{lookForAlignDelims}.

With reference to Table 1 on page 49 and the, yet undisussed, fields of \texttt{noAdditionalIndent} and \texttt{indentRules} (see Section 5.8 on page 48), these comment-marked blocks are considered environments.

### 5.5.3 \texttt{lookForAlignDelims}: the don\texttt{\textunderscore}tMeasure feature

The \texttt{lookForAlignDelims} field can, optionally, receive the \texttt{dontMeasure} option which can be specified in a few different ways. We will explore this feature in relation to the code given in Listing 72; the default output is shown in Listing 73.
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 75, we can run the command

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

and receive the output given in Listing 74.

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

The `dontMeasure` field can also be specified in the form demonstrated in Listing 77. On running the following commands,

```
cmh:~$ latexindent.pl tabular-DM.tex -l=dontMeasure2.yaml
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 78.
5.5 Aligning at delimiters

We note that in:

- Listing 79 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 80 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.

We may also specify the applyTo field as row, a demonstration of which is given in Listing 82; upon running

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

we receive the output in Listing 81.

Finally, the applyTo field can be specified as row, together with a regex expression. For example, for the settings given in Listing 84, upon running

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

we receive the output in Listing 83.

### 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.
5.5 Aligning at delimiters

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

We demonstrate how to customise this with respect to the code given in Listing 85; the default output from latexindent.pl is given in Listing 86.

<table>
<thead>
<tr>
<th>Listing 85: tabbing.tex</th>
<th>Listing 86: tabbing.tex default output</th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{tabbing}</td>
<td>\begin{tabbing}</td>
</tr>
<tr>
<td>aa = bb = cc = dd = ee</td>
<td>aa = bb = cc = dd = ee</td>
</tr>
<tr>
<td>&gt;2 &gt; 1 &gt; 7 &gt; 3</td>
<td>&gt;2 &gt; 1 &gt; 7 &gt; 3</td>
</tr>
<tr>
<td>&gt;3 &gt; 2 &gt;8 &gt; 3</td>
<td>&gt;3 &gt; 2 &gt;8 &gt; 3</td>
</tr>
<tr>
<td>&gt;4 &gt; 2</td>
<td>&gt;4 &gt; 2</td>
</tr>
<tr>
<td>\end{tabbing}</td>
<td>\end{tabbing}</td>
</tr>
</tbody>
</table>

Let’s say that we wish to align the code at either the \= or \>. We employ the settings given in Listing 88 and run the command

cmb:~$ latexindent.pl tabbing.tex -l=delimiterRegEx1.yaml

to receive the output given in Listing 87.

<table>
<thead>
<tr>
<th>Listing 87: tabbing.tex using Listing 88</th>
<th>Listing 88: delimiterRegEx1.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{tabbing}</td>
<td>lookForAlignDelims:</td>
</tr>
<tr>
<td>aa = bb = cc = dd = ee</td>
<td>tabbing:</td>
</tr>
<tr>
<td>&gt; 2 &gt; 1 &gt; 7 &gt; 3</td>
<td>delimiterRegEx: '(...)'</td>
</tr>
<tr>
<td>&gt; 3 &gt; 2 &gt; 8 &gt; 3</td>
<td></td>
</tr>
<tr>
<td>&gt; 4 &gt; 2</td>
<td></td>
</tr>
<tr>
<td>\end{tabbing}</td>
<td></td>
</tr>
</tbody>
</table>

We note that:

- in Listing 87 the code has been aligned, as intended, at both the \= and \>;
- in Listing 88 we have heeded the warning and captured the expression using grouping parenthesis, specified a backslash using \ and said that it must be followed by either \= or \>.

We can explore delimiterRegEx a little further using the settings in Listing 90 and run the command

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

to receive the output given in Listing 89.

<table>
<thead>
<tr>
<th>Listing 89: tabbing.tex using Listing 90</th>
<th>Listing 90: delimiterRegEx2.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{tabbing}</td>
<td>lookForAlignDelims:</td>
</tr>
<tr>
<td>aa = bb = cc = dd = ee</td>
<td>tabbing:</td>
</tr>
<tr>
<td>&gt; 2 &gt; 1 &gt; 7 &gt; 3</td>
<td>delimiterRegEx: '(...)'</td>
</tr>
<tr>
<td>&gt; 3 &gt; 2 &gt; 8 &gt; 3</td>
<td></td>
</tr>
<tr>
<td>&gt; 4 &gt; 2</td>
<td></td>
</tr>
<tr>
<td>\end{tabbing}</td>
<td></td>
</tr>
</tbody>
</table>

We note that only the \> have been aligned.

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 36 on page 31 remain the same; for example, using the settings in Listing 92, and running
5.6 Indent after items, specials and headings

The environment names specified in `indentAfterItems` tell `latexindent.pl` to look for \item commands; if these switches are set to 1 then indentation will be performed so as indent the code after each item. A demonstration is given in Listings 99 and 100.
5.6 Indent after items, specials and headings

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 21 for details of how to configure user settings, and Listing 13 on page 22 in particular.)

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 102 shows the default settings of specialBeginEnd.

The field displayMath represents $ [...] $, inlineMath represents $ ... $ and displayMathTex represents $$ ... $$. You can, of course, rename these in your own YAML files (see Section 4.2 on page 22); indeed, you might like to set up your own special begin and end statements.

A demonstration of the before-and-after results are shown in Listings 103 and 104.
5.6 Indent after items, specials and headings

LISTING 103: special1.tex before
The function \( f \) has formula
\[
\begin{array}{l}
f(x)=x^2.
\end{array}
\]
If you like splitting dollars,
\[
g(x)=f(2x)
\]

LISTING 104: special1.tex default output
The function \( f \) has formula
\[
\begin{array}{l}
f(x)=x^2.
\end{array}
\]
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. For example, consider the file shown in Listing 105.

LISTING 105: specialLR.tex
\begin{equation}
\left[\sqrt{a+b}\right]
\end{equation}

Now consider the YAML files shown in Listings 106 and 107

LISTING 106: specialsLeftRight.yaml
`specialBeginEnd:
  leftRightSquare:
    begin: '\left[\'
    end: '\right]'
  lookForThis: 1`

LISTING 107: specialBeforeCommand.yaml
`specialBeginEnd:
  specialBeforeCommand: 1`

Upon running the following commands

```
cmh:~$ latexindent.pl specialLR.tex -l=specialsLeftRight.yaml
cmh:~$ latexindent.pl specialLR.tex -l=specialsLeftRight.yaml,specialBeforeCommand.yaml
```

we receive the respective outputs in Listings 108 and 109.

LISTING 108: specialLR.tex using Listing 106
\begin{equation}
\left[\sqrt{a+b}\right]
\end{equation}

LISTING 109: specialLR.tex using Listings 106 and 107
\begin{equation}
\left[\sqrt{a+b}\right]
\end{equation}

Notice that in:
- Listing 108 the `\left` has been treated as a `command`, with one optional argument;
- Listing 109 the `specialBeginEnd` pattern in Listing 106 has been obeyed because Listing 107 specifies that the `specialBeginEnd` should be sought before commands.
5.6 Indent after items, specials and headings

You can, optionally, specify the middle field for anything that you specify in specialBeginEnd. For example, let's consider the .tex file in Listing 110.

**LISTING 110: special2.tex**
\begin{verbatim}
\If
  something 0
\ElseIf
  something 1
\ElseIf
  something 2
\ElseIf
  something 3
\Else
  something 4
\EndIf
\end{verbatim}

Upon saving the YAML settings in Listings 111 and 113 and running the commands

```
cmh:~/\$ latexindent.pl special2.tex -l=middle
```
```
cmh:~/\$ latexindent.pl special2.tex -l=middle1
```
then we obtain the output given in Listings 112 and 114.

**LISTING 111: middle.yaml**

```
specialBeginEnd:
  If:
    begin: '\\If'
    middle: '\\ElseIf'
    end: '\\EndIf'
    lookForThis: 1
```

**LISTING 112: special2.tex using Listing 111**
```
\If
  something 0
\ElseIf
  something 1
\ElseIf
  something 2
\ElseIf
  something 3
\Else
  something 4
\EndIf
```

**LISTING 113: middle1.yaml**

```
specialBeginEnd:
  If:
    begin: '\\If'
    middle: - '\\ElseIf'
    middle: - '\\Else'
    end: '\\EndIf'
    lookForThis: 1
```

**LISTING 114: special2.tex using Listing 113**
```
\If
  something 0
\ElseIf
  something 1
\ElseIf
  something 2
\ElseIf
  something 3
\Else
  something 4
\EndIf
```

We note that:

- in Listing 112 the bodies of each of the ElseIf statements have been indented appropriately;
- the Else statement has not been indented appropriately in Listing 112 – read on!
- we have specified multiple settings for the middle field using the syntax demonstrated in Listing 113 so that the body of the Else statement has been indented appropriately in Listing 114.
You may specify fields in `specialBeginEnd` to be treated as verbatim code blocks by changing `lookForThis` to be `verbatim`.

For example, beginning with the code in Listing 116 and the YAML in Listing 115, and running

```
cmh:~$ latexindent.pl special3.tex -l=special-verbi
```

then the output in Listing 116 is unchanged.

```
\[
\text{special code blocks can be treated as verbatim}
\]
```

We can combine the `specialBeginEnd` with the `lookForAlignDelims` feature. We begin with the code in Listing 117.

```
\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 [bend left] 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}
```

Let's assume that our goal is to align the code at the edge and node text; we employ the code given in Listing 118 and run the command

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

to receive the output in Listing 119.

```
\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 [bend left] 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}
```

The output in Listing 119 is not quite ideal. We can tweak the settings within Listing 118 in order to improve the output; in particular, we employ the code in Listing 120 and run the command

```
\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 [bend left] 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}
```
to receive the output in Listing 121.

\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 [bend left] node {1,0,R} (E)
(D) edge [loop below] node {1,1,R} (D)
edge node {0,1,R} (A)
\end{tikzpicture}

The lookForThis field can be considered optional; by default, it is assumed to be 1, which is demonstrated in Listing 120.

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.\footnote{There is a slight difference in interface for this field when comparing Version 2.2 to Version 3.0; see appendix F on page 143 for details.}

\begin{lstlisting}[language=yaml]
indentAfterHeadings: (fields)
\end{lstlisting}

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 deep.

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 page 48); you will find the default indentRules contains chapter: " " which tells latexindent.pl simply to use a space character after headings (once indent is set to 1 for chapter).

For example, assuming that you have the code in Listing 123 saved into headings1.yaml, and that you have the text from Listing 124 saved into headings1.tex.
5.6 Indent after items, specials and headings

<table>
<thead>
<tr>
<th>LISTING 123: headings1.yaml</th>
<th>LISTING 124: headings1.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>indentAfterHeadings:</td>
<td>\subsection{subsection title}</td>
</tr>
<tr>
<td>subsection:</td>
<td>subsection text</td>
</tr>
<tr>
<td>indentAfterThisHeading: 1</td>
<td>subsection text</td>
</tr>
<tr>
<td>level: 1</td>
<td>\paragraph{paragraph title}</td>
</tr>
<tr>
<td>paragraph:</td>
<td>paragraph text</td>
</tr>
<tr>
<td>indentAfterThisHeading: 1</td>
<td>\paragraph{paragraph title}</td>
</tr>
<tr>
<td>level: 2</td>
<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 125.

<table>
<thead>
<tr>
<th>LISTING 125: headings1.tex using Listing 123</th>
</tr>
</thead>
<tbody>
<tr>
<td>\subsection{subsection title}</td>
</tr>
<tr>
<td>\hspace*{1.0em}subsection text</td>
</tr>
<tr>
<td>\hspace*{1.0em}subsection text</td>
</tr>
<tr>
<td>\hspace*{1.0em}\begin{verbatim} \paragraph{paragraph title} \end{verbatim}</td>
</tr>
<tr>
<td>\hspace*{1.0em}\hspace*{1.0em}paragraph text</td>
</tr>
<tr>
<td>\hspace*{1.0em}\hspace*{1.0em}paragraph text</td>
</tr>
<tr>
<td>\hspace*{1.0em}\hspace*{1.0em}\begin{verbatim} \paragraph{paragraph title} \end{verbatim}</td>
</tr>
<tr>
<td>\hspace*{1.0em}\hspace*{1.0em}paragraph text</td>
</tr>
<tr>
<td>\hspace*{1.0em}\hspace*{1.0em}paragraph text</td>
</tr>
</tbody>
</table>

Now say that you modify the YAML from Listing 123 so that the paragraph level is 1; after running

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

you should receive the code given in Listing 126; 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 [25], and is off by default.

For example, consider the example shown in Listing 127 together with the default output shown in Listing 128.
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 1.

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 1 is discussed in Section 8 on page 127.

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 1 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);
2. indentRules for the name of the current (thing);
### Table 1: 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><code>a-zA-Z\*0-9_\</code></td>
<td>\begin{myenv} body of myenv \end{myenv}</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><code>+a-zA-Z\*0-9_\</code></td>
<td>\mycommand{arguments}</td>
</tr>
<tr>
<td>keyEqualsValuesBracesBrackets</td>
<td><code>a-zA-Z\*0-9_/.\h\{\}\#-</code></td>
<td>my key/.style={arguments}</td>
</tr>
<tr>
<td>namedGroupingBracesBrackets</td>
<td><code>0-9/.a-zA-Z\*&gt;&lt;</code></td>
<td>in{arguments}</td>
</tr>
<tr>
<td>UnNamedGroupingBracesBrackets</td>
<td>No name!</td>
<td>{ or [ or , or &amp; or ) or ( or $ followed by (arguments)</td>
</tr>
<tr>
<td>ifElseFi</td>
<td>@a-zA-Z but must begin with either \if of @if \else \fi</td>
<td>\ifnum... ...</td>
</tr>
<tr>
<td>items</td>
<td>User specified, see Listings 98 and 101 on page 42</td>
<td>\begin{enumerate} \item ... \end{enumerate}</td>
</tr>
<tr>
<td>specialBeginEnd</td>
<td>User specified, see Listing 102 on page 42</td>
<td>[ ... ]</td>
</tr>
<tr>
<td>afterHeading</td>
<td>User specified, see Listing 122 on page 46</td>
<td>\chapter{title} ... \section{title}</td>
</tr>
<tr>
<td>filecontents</td>
<td>User specified, see Listing 28 on page 29</td>
<td>\begin{filecontents} ... \end{filecontents}</td>
</tr>
</tbody>
</table>
3. noAdditionalIndentGlobal for the type of the current (thing);
4. indentRulesGlobal for the type of the current (thing).

Using the above list, the first piece of information to be found will be used; failing that, the value of 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 indentRules and in noAdditionalIndentGlobal, then the information from indentRules takes priority.

We now present details for the different type of code blocks known to latexindent.pl, as detailed in Table 1 on the preceding page; for reference, there follows a list of the code blocks covered.

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 131.

**Listing 131: myenv.tex**

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

If we do not wish myenv to receive any additional indentation, we have a few choices available to us, as demonstrated in Listings 132 and 133.

**Listing 132: myenv-noAdd1.yaml**

```yaml
noAdditionalIndent: (fields)
myenv: 1
```

**Listing 133: myenv-noAdd2.yaml**

```yaml
noAdditionalIndent: (fields)
myenv:
  body: 1
```

On applying either of the following commands,
we obtain the output given in Listing 134; note in particular that the environment myenv has not received any additional indentation, but that the outer environment has still received indentation.

**Listing 134: myenv.tex output (using either Listing 132 or Listing 133)**

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

Upon changing the YAML files to those shown in Listings 135 and 136, and running either

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

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

we obtain the output given in Listing 137.

**Listing 135: myenv-noAdd3.yaml**

```
noAdditionalIndent:
  myenv: 0
```

**Listing 136: myenv-noAdd4.yaml**

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

**Listing 137: myenv.tex output (using either Listing 135 or Listing 136)**

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

Let’s now allow myenv to have some optional and mandatory arguments, as in Listing 138.

**Listing 138: myenv-args.tex**

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

Upon running
we obtain the output shown in Listing 139; 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 132), then all parts of the environment (body, optional and mandatory arguments) are assumed to want no additional indent.

We may customise noAdditionalIndent for optional and mandatory arguments of the myenv environment, as shown in, for example, Listings 140 and 141.

Upon running

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

we obtain the respective outputs given in Listings 142 and 143. Note that in Listing 142 the text for the optional argument has not received any additional indentation, and that in Listing 143 the mandatory argument has not received any additional indentation; in both cases, the body has not received any additional indentation.
We may also specify indentation rules for environment code blocks using the `indentRules` field; see, for example, Listings 144 and 145.

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 146; note in particular that the environment `myenv` has received one tab (from the outer environment) plus three spaces from Listing 144 or 145.

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

Returning to the example in Listing 138 that contains optional and mandatory arguments. Upon using Listing 144 as in

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

we obtain the output in Listing 147; note that the body, optional argument and mandatory argument of `myenv` have all received the same customised indentation.

You can specify different indentation rules for the different features using, for example, Listings 148 and 149.
After running

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

and

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

then we obtain the respective outputs given in Listings 150 and 151.

<table>
<thead>
<tr>
<th>Listing 150: myenv-rules3.yaml</th>
<th>Listing 151: myenv-rules4.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>indentRules:</td>
<td>indentRules:</td>
</tr>
<tr>
<td>myenv:</td>
<td>myenv:</td>
</tr>
<tr>
<td>body: &quot; &quot;</td>
<td>body: &quot; &quot;</td>
</tr>
<tr>
<td>optionalArguments: &quot; &quot;</td>
<td>mandatoryArguments: &quot;\t\t&quot;</td>
</tr>
</tbody>
</table>

\begin{outer}
  \begin{myenv}
  \"\text{\textoptionargtext}\
  \"\text{\textoptionargtext}\
  \"\text{\textmandatoryargtext}\
  \"\text{\textmandatoryargtext}\
  \text{\textbodyfromenvironment}\
  \text{\textbodyfromenvironment}\
  \text{\textbodyfromenvironment}\
  \end{myenv}
\end{outer}

\begin{outer}
  \begin{myenv}
  \"\text{\textoptionargtext}\
  \"\text{\textoptionargtext}\
  \"\text{\textmandatoryargtext}\
  \"\text{\textmandatoryargtext}\
  \text{\textbodyfromenvironment}\
  \text{\textbodyfromenvironment}\
  \text{\textbodyfromenvironment}\
  \end{myenv}
\end{outer}

Note that in Listing 150, 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 151, 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.

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 152). Let’s say that you change the value of environments to 1 in Listing 152, and that you run

\begin{verbatim}
cmh:~$ latexindent.pl myenv-args.tex -l env-noAdditionalGlobal.yaml

and

\begin{verbatim}
cmh:~$ latexindent.pl myenv-args.tex -l myenv-rules1.yaml,env-noAdditionalGlobal.yaml
\end{verbatim}
\end{verbatim}

The respective output from these two commands are in Listings 153 and 154; in Listing 153 notice that both environments receive no additional indentation but that the arguments of myenv still do receive indentation. In Listing 154 notice that the outer environment does not receive additional indentation, but because of the settings from myenv-rules1.yaml (in Listing 144 on the previous page), the myenv environment still does receive indentation.
In fact, noAdditionalIndentGlobal also contains keys that control the indentation of optional and mandatory arguments; on referencing Listings 155 and 156

we may run the commands

```bash
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 157 and 158. Notice that in Listing 157 the optional argument has not received any additional indentation, and in Listing 158 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}
```

The final check that latexindent.pl will make is to look for indentRulesGlobal as detailed in Listing 159; if you change the environments field to anything involving horizontal space, say \"\\", and then run the following commands

```bash
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 160 and 161. Note that in Listing 160, both the environment blocks have received a single-space indentation, whereas in Listing 161 the outer environment has received single-space indentation (specified by indentRulesGlobal), but myenv has received \"\", as specified by the particular indentRules for myenv Listing 144 on page 53.

You can specify indentRulesGlobal for both optional and mandatory arguments, as detailed in Listings 162 and 163.

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 164 and 165. Note that the optional argument in Listing 164 has received two tabs worth of indentation, while the mandatory argument has done so in Listing 165.

5.8.2 Environments with items

With reference to Listings 98 and 101 on page 42, some commands may contain item commands; for the purposes of this discussion, we will use the code from Listing 99 on page 42.

Assuming that you’ve populated itemNames with the name of your item, you can put the item name into noAdditionalIndent as in Listing 166, although a more efficient approach may be to change the relevant field in itemNames to 0. Similarly, you can customise the indentation that your item receives using indentRules, as in Listing 167.
Upon running the following commands

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

Alternatively, you might like to populate noAdditionalIndentGlobal or indentRulesGlobal using the items key, as demonstrated in Listings 170 and 171. Note that there is a need to ‘reset/remove’ the item field from indentRules in both cases (see the hierarchy description given on page 48) as the item command is a member of indentRules by default.

Upon running the following commands,

\begin{itemize}
  \item some\_text\_here
  some\_more\_text\_here
  another\_item
  some\_more\_text\_here
\end{itemize}

the respective outputs from Listings 168 and 169 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

Let’s begin with the simple example in Listing 172; when latexindent.pl operates on this file, the default output is shown in Listing 173.  

\footnote{The command code blocks have quite a few subtleties, described in Section 5.9 on page 66.}
As in the environment-based case (see Listings 132 and 133 on page 50) we may specify `noAdditionalIndent` either in ‘scalar’ form, or in ‘field’ form, as shown in Listings 174 and 175.

Listing 174:
```
mycommand
{ 
mand arg text
mand arg text}
[ 
opt arg text
opt arg text
] 
```

Listing 175:
```
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 176 and 177.

Listing 176:
```
\mycommand
{ 
mand arg text
mand arg text}
[ 
opt arg text
opt arg text
] 
```

Listing 177:
```
\mycommand
{ 
mand arg text
mand arg text}
[ 
opt arg text
opt arg text
] 
```

Note that in Listing 176 that the ‘body’, optional argument and mandatory argument have all received no additional indentation, while in Listing 177, 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.

We may further customise `noAdditionalIndent` for `mycommand` as we did in Listings 140 and 141 on page 52; explicit examples are given in Listings 178 and 179.

Listing 178:
```
noAdditionalIndent:
mycommand:
body: 0
optionalArguments: 1
mandatoryArguments: 0
```

Listing 179:
```
noAdditionalIndent:
mycommand:
body: 0
optionalArguments: 0
mandatoryArguments: 1
```
we receive the respective output given in Listings 180 and 181.

**Listing 180:** mycommand.tex using Listing 178
```
\mycommand
{  
  mand arg text
  mand arg text}
[
  opt arg text
  opt arg text
]
```

**Listing 181:** mycommand.tex using Listing 179
```
\mycommand
{  
  mand arg text
  mand arg text}
[
  opt arg text
  opt arg text
]
```

Attentive readers will note that the body of mycommand in both Listings 180 and 181 has received no additional indent, even though body is explicitly set to 0 in both Listings 178 and 179. This is because, by default, noAdditionalIndentGlobal for commands is set to 1 by default; this can be easily fixed as in Listings 182 and 183.

**Listing 182:** mycommand-noAdd5.yaml
```
noAdditionalIndent:
  mycommand:
    body: 0
    optionalArguments: 1
    mandatoryArguments: 0
  noAdditionalIndentGlobal:
    commands: 0
```

**Listing 183:** mycommand-noAdd6.yaml
```
noAdditionalIndent:
  mycommand:
    body: 0
    optionalArguments: 0
    mandatoryArguments: 1
  noAdditionalIndentGlobal:
    commands: 0
```

After running the following commands,

```
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 184 and 185.

**Listing 184:** mycommand.tex using Listing 182
```
\mycommand
{  
  mand arg text
  mand arg text}
[
  opt arg text
  opt arg text
]
```

**Listing 185:** mycommand.tex using Listing 183
```
\mycommand
{  
  mand arg text
  mand arg text}
[
  opt arg text
  opt arg text
]
```

Both indentRules and indentRulesGlobal can be adjusted as they were for environment code blocks, as in Listings 148 and 149 on page 54 and Listings 159, 162 and 163 on pages 55–56.

### 5.8.4 ifelsefi code blocks

Let’s use the simple example shown in Listing 186; when latexindent.pl operates on this file, the output as in Listing 187; note that the body of each of the \if statements have been indented, and that the \else statement has been accounted for correctly.
It is recommended to specify \texttt{noAdditionalIndent} and \texttt{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 188 and 189.

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 190 and 191; note that in Listing 190, the \texttt{ifnum} code block has not received any additional indentation, while in Listing 191, the \texttt{ifnum} code block has received one tab and two spaces of indentation.

We may specify \texttt{noAdditionalIndentGlobal} and \texttt{indentRulesGlobal} as in Listings 192 and 193.

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 194 and 195; notice that in Listing 194 neither of the \texttt{ifelsefi} code blocks have received indentation, while in Listing 195 both code blocks have received a single space of indentation.
We can further explore the treatment of `ifElseFi` code blocks in Listing 196, and the associated default output given in Listing 197; note, in particular, that the bodies of each of the 'or statements' have been indented.

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

5.8.5 specialBeginEnd code blocks

Let's use the example from Listing 103 on page 43 which has default output shown in Listing 104 on page 43.

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 198 and 199.

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

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

After running the following commands,

```bash
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 200 and 201; note that in Listing 200, the `displayMath` code block has not received any additional indentation, while in Listing 201, the `displayMath` code block has received three tabs worth of indentation.
The function $f$ has formula
\[
\begin{align*}
f(x) &= x^2. \\
\end{align*}
\]
If you like splitting dollars,
\[
\begin{align*}
g(x) &= f(2x) \\
\end{align*}
\]
We may specify noAdditionalIndentGlobal and indentRulesGlobal as in Listings 202 and 203. 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 204 and 205; notice that in Listing 204 neither of the special code blocks have received indentation, while in Listing 205 both code blocks have received a single space of indentation.

5.8.6 afterHeading code blocks

Let's use the example Listing 206 for demonstration throughout this Section. As discussed on page 47, by default latexindent.pl will not add indentation after headings.

On using the YAML file in Listing 208 by running the command

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

we obtain the output in Listing 207. Note that the argument of paragraph has received (default) indentation, and that the body after the heading statement has received (default) indentation.
If we specify noAdditionalIndent as in Listing 210 and run the command

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

then we receive the output in Listing 209. 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.

Similarly, if we specify indentRules as in Listing 212 and run analogous commands to those above, we receive the output in Listing 211; note that the body, mandatory argument and content after the heading of paragraph have all received three tabs worth of indentation.

We may, instead, specify noAdditionalIndent in ‘field’ form, as in Listing 214 which gives the output in Listing 213.

Analogously, we may specify indentRules as in Listing 216 which gives the output in Listing 215; 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.
Finally, let’s consider noAdditionalIndentGlobal and indentRulesGlobal shown in Listings 218 and 220 respectively, with respective output in Listings 217 and 219. Note that in Listing 218 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 219, the argument has received both a (default) tab plus two spaces of indentation (from the global rule specified in Listing 220), and the remaining body after paragraph has received just two spaces of indentation.

5.8.7 The remaining code blocks

Referencing the different types of code blocks in Table 1 on page 49, we have a few code blocks yet to cover; these are very similar to the commands code block type covered comprehensively in Section 5.8.3 on page 57, 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 1 on page 49;
- then an = symbol;
- then at least one set of curly braces or square brackets (comments and line breaks allowed throughout).

See the keyEqualsValuesBracesBrackets: follow and keyEqualsValuesBracesBrackets: name fields of the fine tuning section in Listing 502 on page 127

An example is shown in Listing 221, with the default output given in Listing 222.
In Listing 222, note that the maximum indentation is three tabs, and these come from:

- the `\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 `body` field for `noAdditionalIndent` and friends from page 48.

### 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 1 on page 49;
- 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 502 on page 127

A simple example is given in Listing 223, with default output in Listing 224.

**Listing 223: child1.tex**
```
\coordinate 
child[grow=down]{
  edge from parent [antiparticle]
  node [above=3pt] {\$C\$}
}
```
**Listing 224: child1.tex default output**
```
\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`\(^7\). Referencing Listing 224, 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 48.

### 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 `{` 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 502 on page 127

An example is shown in Listing 225 with default output give in Listing 226.

**Listing 225: psforeach1.tex**
```
\psforeach {%
  \row
  \{%
    {3,2.8,2.7,3,3.1},% 
    {2.8,1,1.2,2,3},%
  \}
}\psforeach \row {%
  %
 \{%
    {3,2.8,2.7,3,3.1},%
    {2.8,1,1.2,2,3},%
  \}
```

\(^7\)You may like to verify this by using the `-tt` option and checking `indent.log`!
Referencing Listing 226, there are three sets of unnamed braces. Note also that the maximum value of indentation is three tabs, and these come from:

- the \psforeach command’s mandatory argument;
- the first un-named braces mandatory argument;
- the first un-named braces body, which we define as any lines following the first opening { or [ that defined the code block. This is the part controlled by the body field for noAdditionalIndent and friends from page 48.

Users wishing to customise the mandatory and/or optional arguments on a per-name basis for the UnNamedGroupingBracesBrackets should use always-un-named.

5.8.7.4 filecontents

code blocks behave just as environments, except that neither arguments nor items are sought.

5.8.8 Summary

Having considered all of the different types of code blocks, the functions of the fields given in Listings 227 and 228 should now make sense.

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 229.
5.9 Commands and the strings between their arguments

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 230.

Listing 230: pstricks1.tex
```
defFunction[algebraic]{torus}(u,v)
{(2+cos(u))*cos(v+\Pi)}
{(2+cos(u))*sin(v+\Pi)}
{sin(u)}
```

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 229, then \latexindent.pl will allow round parenthesis between optional and mandatory arguments. In the case of the code in Listing 230, \latexindent.pl finds all the arguments of \defFunction, both before and after \((u,v)\).

The default output from running \latexindent.pl on Listing 230 actually leaves it unchanged (see Listing 231); note in particular, this is because of noAdditionalIndentGlobal as discussed on page 59.

Upon using the YAML settings in Listing 233, and running the command
```
$ latexindent.pl pstricks1.tex -l noRoundParentheses.yaml
```
we obtain the output given in Listing 232.

Notice the difference between Listing 231 and Listing 232; in particular, in Listing 232, 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 1 on page 49) which, by default, assume indentation for their body, and hence the tabbed indentation in Listing 232.

Let’s explore this using the YAML given in Listing 235 and run the command
```
```

5.9 Commands and the strings between their arguments

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 230.

Listing 230: pstricks1.tex
```
defFunction[algebraic]{torus}(u,v)
{(2+cos(u))*cos(v+\Pi)}
{(2+cos(u))*sin(v+\Pi)}
{sin(u)}
```

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 229, then \latexindent.pl will allow round parenthesis between optional and mandatory arguments. In the case of the code in Listing 230, \latexindent.pl finds all the arguments of \defFunction, both before and after \((u,v)\).

The default output from running \latexindent.pl on Listing 230 actually leaves it unchanged (see Listing 231); note in particular, this is because of noAdditionalIndentGlobal as discussed on page 59.

Upon using the YAML settings in Listing 233, and running the command
```
$ latexindent.pl pstricks1.tex -l noRoundParentheses.yaml
```
we obtain the output given in Listing 232.

Notice the difference between Listing 231 and Listing 232; in particular, in Listing 232, 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 1 on page 49) which, by default, assume indentation for their body, and hence the tabbed indentation in Listing 232.

Let’s explore this using the YAML given in Listing 235 and run the command
5.9 Commands and the strings between their arguments

then the output is as in Listing 234.

```
\defFunction[algebraic]{torus}(u,v)
  \{(2+\cos(u))\cos(v+\Pi)\}
\{(2+\cos(u))\sin(v+\Pi)\}
\{\sin(u)\}
```

Notice in Listing 234 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 235.

```
stringsAllowedBetweenArguments: (fields)
```

---

tikz users may well specify code such as that given in Listing 236; processing this code using latexindent.pl gives the default output in Listing 237.

```
\draw[thin]
(c)\_{\text{to}}[\text{in}=110,\text{out}=-90]
++(0,-0.5cm)
node[\text{below,align}=\text{left, scale}=0.5]
```

With reference to Listing 229 on the previous 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 236, 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]

We can explore this further, for example using Listing 239 and running the command

```
~\$ latexindent.pl tikz-node1.tex -l draw.yaml
```

we receive the output given in Listing 238.
5.9 Commands and the strings between their arguments

Notice that each line after the `\draw` command (its ‘body’) in Listing 238 has been given the appropriate two-spaces worth of indentation specified in Listing 239.

Let's compare this with the output from using the YAML settings in Listing 241, and running the command

```
cmh:~$ latexindent.pl tikz-node1.tex -l no-strings.yaml
```

given in Listing 240.

In this case, 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 1 on page 49) 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 229 on page 67, 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 242 or Listing 243 is equivalent to using the settings in Listing 244.

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 245 means that only the strings specified in that field will be used.
5.9 Commands and the strings between their arguments

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 243 to 245.

We may explore this feature further with the code in Listing 246, whose default output is given in Listing 247.

```
\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 249, and running the command

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

given in Listing 248.

```
\foreach \x/\y in \{0/1,1/2\}{
body of foreach
}
```

You might like to compare the output given in Listing 247 and Listing 248. Note, in particular, in Listing 247 that the `foreach` command has not included any of the subsequent strings, and that the braces have been treated as a `namedGroupingBracesBrackets`. In Listing 248 the `foreach` command has been allowed to have `\x/\y` and in between arguments because of the settings given in Listing 249.

```
commandNameSpecial: \{fields\}
```

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 ].

For example, consider the sample file in Listing 250, which has default output in Listing 251.

```
\parbox{
@ifnextchar\{arg 1\}{arg 2}
}
```

Notice that in Listing 251 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 <thing> and works outwards*, discussed in more detail on page 116.
For demonstration, we can compare this output with that given in Listing 252 in which the settings from Listing 253 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.

<table>
<thead>
<tr>
<th>Listing 252: <code>ifnextchar.tex</code> using <code>Listing 253</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>\parbox{ @ifnextchar[{arg 1}{arg 2} }</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 253: <code>no-ifnextchar.yaml</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>commandCodeBlocks:</td>
</tr>
<tr>
<td>commandNameSpecial: 0</td>
</tr>
</tbody>
</table>

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

⚠️ 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 243 to 245.
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 .......................................................... 73
   6.1.1 Text wrap quick start ............................................. 74
   6.1.2 textWrapOptions: modifying line breaks by text wrapping ... 74
   6.1.3 Text wrapping on a per-code-block basis .................... 77
6.2 removeParagraphLineBreaks: modifying line breaks for paragraphs .......... 83
6.3 Combining removeParagraphLineBreaks and textWrapOptions .................. 89
   6.3.1 text wrapping beforeFindingChildCodeBlocks ................ 90
6.4 Summary of text wrapping .......................................... 92
6.5 oneSentencePerLine: modifying line breaks for sentences .................... 92
   6.5.1 sentencesFollow ................................................ 94
   6.5.2 sentencesBeginWith ............................................ 95
   6.5.3 sentencesEndWith ............................................. 96
   6.5.4 Features of the oneSentencePerLine routine .................... 98
   6.5.5 Text wrapping and indenting sentences ........................ 99
6.6 Poly-switches ......................................................... 101
   6.6.1 Poly-switches for environments .................................. 101
      6.6.1.1 Adding line breaks: BeginStartsOnOwnLine and BodyStartsOnOwn-Line .... 102
      6.6.1.2 Adding line breaks using EndStartsOnOwnLine and EndFinishesWith-LineBreak ... 104
      6.6.1.3 poly-switches 1, 2, and 3 only add line breaks when necessary ... 105
      6.6.1.4 Removing line breaks (poly-switches set to −1) .................. 106
      6.6.1.5 About trailing horizontal space ................................ 107
      6.6.1.6 poly-switch line break removal and blank lines ............... 107
   6.6.2 Poly-switches for double back slash ................................ 109
      6.6.2.1 Double back slash starts on own line ....................... 109
      6.6.2.2 Double back slash finishes with line break .................. 110
      6.6.2.3 Double back slash poly-switches for specialBeginEnd ............ 110
      6.6.2.4 Double back slash poly-switches for optional and mandatory arguments 111
      6.6.2.5 Double back slash optional square brackets ................... 112
   6.6.3 Poly-switches for other code blocks ................................ 112
   6.6.4 Partnering BodyStartsOnOwnLine with argument-based poly-switches 114
As of Version 3.0, latexindent.pl has the \texttt{-m} switch, which permits 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 254.

Having read the previous paragraph, it should sound reasonable that, if you call 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{center}
\textbf{If you call 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{center}

\begin{itemize}
  \item \texttt{preserveBlankLines: 0|1}
  \item \texttt{condenseMultipleBlankLinesInto: \{positive integer\}}
\end{itemize}

This field is directly related to \textit{poly-switches}, discussed in Section 6.6. By default, it is set to 1, which means that blank lines will be protected from removal; however, regardless of this setting, multiple blank lines can be condensed if \texttt{condenseMultipleBlankLinesInto} is greater than 0, discussed next.

Assuming that this switch takes an integer value greater than 0, latexindent.pl will condense multiple blank lines into the number of blank lines illustrated by this switch. As an example, Listing 255 shows a sample file with blank lines; upon running

\begin{verbatim}
cmh:$ latexindent.pl myfile.tex -m -o=-mod1
\end{verbatim}

the output is shown in Listing 256; note that the multiple blank lines have been condensed into one blank line, and note also that we have used the \texttt{-m} switch!

\begin{verbatim}
LISTING 255: mlb1.tex
before blank line
after blank line
after blank line
\end{verbatim}

\begin{verbatim}
LISTING 256: mlb1-mod1.tex
before blank line
after blank line
after blank line
\end{verbatim}

\section{Text Wrapping}

There are many different configuration options for the text wrapping routine of latexindent.pl, perhaps too many. The following sections are comprehensive, but quite long; in an attempt to to be brief, you might begin with the settings given in Section 6.1.1.
6.1 Text Wrapping

6.1.1 Text wrap quick start

Of all the available text wrapping options, I consider Listing 257 to be among the most helpful starting points.

```yaml
modifyLineBreaks:
  textWrapOptions:
    columns: 80  # number of columns
    perCodeBlockBasis: 1  # per-code-block wrap
    beforeFindingChildCodeBlocks: 1  # wrap *before* finding child code blocks
    masterDocument: 1  # apply to main document
    afterHeading: 1  # after headings
    items: 1  # within items
    removeParagraphLineBreaks:  # remove line breaks within paragraphs
      masterDocument: 1
      afterHeading: 1
      items: 1
    beforeTextWrap: 1  # before wrapping text
```

You can read about `perCodeBlockBasis` in Section 6.1.3 and `removeParagraphLineBreaks` in Section 6.2.

If the settings in Listing 257 do not give your desired output, take a look at the demonstration in Section 6.3.1, in particular Listing 324.

6.1.2 `textWrapOptions`: modifying line breaks by text wrapping

When the `-m` switch is active `latexindent.pl` has the ability to wrap text using the options specified in the `textWrapOptions` field, see Listing 258.

```yaml
513 textWrapOptions:
514   columns: 0
```

The value of `columns` specifies the column at which the text should be wrapped.

By default, the value of `columns` is 0, so `latexindent.pl` will not wrap text; if you change it to a value of 2 or more, then text will be wrapped after the character in the specified column.

By default, the text wrapping routine will operate before the code blocks have been searched for; text wrapping on a `per-code-block` basis is discussed in Section 6.1.3.

We consider the file given in Listing 259 for demonstration.

```latex
Here is a line of text that will be wrapped by latexindent.pl. Each line is quite long.

Here is a line of text that will be wrapped by latexindent.pl. Each line is quite long.
```

Using the file `textwrap1.yaml` in Listing 261, and running the command

```
cmb:~$ latexindent.pl -m textwrap1.tex -o textwrap1-mod1.tex -l textwrap1.yaml
```

we obtain the output in Listing 260.
6.1 Text Wrapping

LISTING 260: textwrap1-mod1.tex

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

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

The text wrapping routine is performed after verbatim environments have been stored, so verbatim environments and verbatim commands are exempt from the routine. For example, using the file in Listing 262,

LISTING 262: textwrap2.tex

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

\begin{verbatim}
 a long line in a verbatim environment, which will not be broken by latexindent.pl
\end{verbatim}

Here is a verb command: \verb!this will not be text wrapped!

and running the following command and continuing to use textwrap1.yaml from Listing 261,

\texttt{cmh:~$ latexindent.pl -m textwrap2.tex -o textwrap2-mod1.tex -l textwrap1.yaml}

then the output is as in Listing 263.

LISTING 263: textwrap2-mod1.tex

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

\begin{verbatim}
 a long line in a verbatim environment, which will not be broken by latexindent.pl
\end{verbatim}

Here is a verb command: \verb!this will not be text wrapped!

Furthermore, the text wrapping routine is performed after the trailing comments have been stored, and they are also exempt from text wrapping. For example, using the file in Listing 264
Here is a line of text that will be wrapped by \textindent{pl}. Each line is quite long.

and running the following command and continuing to use \text\yaml{1} from Listing 261,

\begin{verbatim}
  \$ latexindent\ .pl -m textwrap3.tex -o textwrap3-mod1.tex -1 textwrap1.yaml
\end{verbatim}

then the output is as in Listing 265.

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 \cite{26}. 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:

\begin{itemize}
  \item Changing the value of huge to anything other than overflow will slow down \textindent{pl} significantly when the -m switch is active.
  \item 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.
\end{itemize}

For example, using the settings in Listings 267 and 269 and running the commands

\begin{verbatim}
  \$ latexindent\ .pl -m textwrap4.tex -o=-mod2A -1 textwrap2A.yaml
  \$ latexindent\ .pl -m textwrap4.tex -o=-mod2B -1 textwrap2B.yaml
\end{verbatim}

gives the respective output in Listings 266 and 268.

\begin{verbatim}
  >>> latexindent -m texwrap4-mod2B concurrently
  >>> latexindent -m texwrap4-mod2B concurrently
\end{verbatim}
6.1 Text Wrapping

Here is a line of text.

You can also specify the tabstop field as an integer value, which is passed to the text wrap module; see [26] for details. Starting with the code in Listing 270 with settings in Listing 271, and running the command

```bash
$ latexindent -m textwrap-ts.tex -o=+-mod1 -l tabstop.yaml
```

gives the code given in Listing 272.

6.1.3 Text wrapping on a per-code-block basis

By default, if the value of `columns` is greater than 0 and the `-m` switch is active, then the text wrapping routine will operate before the code blocks have been searched for. This behaviour is customisable; in particular, you can instead instruct `latexindent.pl` to apply `textWrap` on a per-code-block basis. Thanks to [30] for their help in testing and shaping this feature.

The full details of `textWrapOptions` are shown in Listing 273. In particular, note the field `perCodeBlockBasis`: 0.

The code blocks detailed in Listing 273 are with direct reference to those detailed in Table 1 on page 49.
The only special case is the *masterDocument* field; this is designed for 'chapter'-type files that may contain paragraphs that are not within any other code-blocks. The same notation is used between this feature and the *removeParagraphLineBreaks* described in Listing 292 on page 83; in fact, the two features can even be combined (this is detailed in Section 6.3 on page 89).

Let’s explore these switches with reference to the code given in Listing 274; the text outside of the environment is considered part of the *masterDocument*.

<table>
<thead>
<tr>
<th>Listing 274: textwrap5.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before the environment; here is a line of text that can be wrapped by latexindent.pl.</td>
</tr>
<tr>
<td>\begin{myenv}</td>
</tr>
<tr>
<td>Within the environment; here is a line of text that can be wrapped by latexindent.pl.</td>
</tr>
<tr>
<td>\end{myenv}</td>
</tr>
<tr>
<td>After the environment; here is a line of text that can be wrapped by latexindent.pl.</td>
</tr>
</tbody>
</table>

With reference to this code block, the settings given in Listings 275 to 277 each give the same output.

<table>
<thead>
<tr>
<th>Listing 275: textwrap3.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>modifyLineBreaks:</td>
</tr>
<tr>
<td>textWrapOptions:</td>
</tr>
<tr>
<td>columns: 30</td>
</tr>
<tr>
<td>perCodeBlockBasis: 1</td>
</tr>
<tr>
<td>all: 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 276: textwrap4.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>modifyLineBreaks:</td>
</tr>
<tr>
<td>textWrapOptions:</td>
</tr>
<tr>
<td>columns: 30</td>
</tr>
<tr>
<td>perCodeBlockBasis: 1</td>
</tr>
<tr>
<td>environments: 1</td>
</tr>
<tr>
<td>masterDocument: 1</td>
</tr>
</tbody>
</table>

Let’s explore the similarities and differences in the equivalent (with respect to Listing 274) syntax specified in Listings 275 to 277:

- in each of Listings 275 to 277 notice that *columns: 30*;
- in each of Listings 275 to 277 notice that *perCodeBlockBasis: 1*;
- in Listing 275 we have specified *all: 1* so that the text wrapping will operate upon all code blocks;
- in Listing 276 we have not specified all, and instead, have specified that text wrapping should be applied to each of environments and masterDocument;
- in Listing 277 we have specified text wrapping for masterDocument and on a per-name basis for environments code blocks.

Upon running the following commands

```
cmh:~$ latexindent.pl -s textwrap5.tex -l=textwrap3.yaml -m
```

we obtain the output shown in Listing 278.
6.1 Text Wrapping

Before the environment; here is a line of text that can be wrapped by latexindent.pl.

\begin{myenv}
   Within the environment; here is a line of text that can be wrapped by latexindent.pl.
\end{myenv}

After the environment; here is a line of text that can be wrapped by latexindent.pl.

We can explore the idea of per-name text wrapping given in Listing 277 by using Listing 279.

Before the environment; here is a line of text that can be wrapped by latexindent.pl.

\begin{myenv}
   Within the environment; here is a line of text that can be wrapped by latexindent.pl.
\end{myenv}

\begin{another}
   Within the environment; here is a line of text that can be wrapped by latexindent.pl.
\end{another}

After the environment; here is a line of text that can be wrapped by latexindent.pl.

In particular, upon running

```
cmh:~$ latexindent.pl -s textwrap6.tex -l=textwrap5.yaml -m
```

we obtain the output given in Listing 280.

Before the environment; here is a line of text that can be wrapped by latexindent.pl.

\begin{myenv}
   Within the environment; here is a line of text that can be wrapped by latexindent.pl.
\end{myenv}

\begin{another}
   Within the environment; here is a line of text that can be wrapped by latexindent.pl.
\end{another}

After the environment; here is a line of text that can be wrapped by latexindent.pl.

Notice that, because environments has been specified only for myenv (in Listing 277) that the environment named another has not had text wrapping applied to it.
The all field can be specified with exceptions which can either be done on a per-code-block or per-name basis; we explore this in relation to Listing 279 in the settings given in Listings 281 to 283.

Upon running the commands

```
cmh:~$ latexindent.pl -s textwrap6.tex -l=textwrap6.yaml -m
cmh:~$ latexindent.pl -s textwrap6.tex -l=textwrap7.yaml -m
cmh:~$ latexindent.pl -s textwrap6.tex -l=textwrap8.yaml -m
```

we receive the respective output given in Listings 284 to 286.

**Listing 281: textwrap6.yaml**

```yaml
modifyLineBreaks:
  textWrapOptions:
    columns: 30
    perCodeBlockBasis: 1
    all:
      except:
        - environments
```

**Listing 282: textwrap7.yaml**

```yaml
modifyLineBreaks:
  textWrapOptions:
    columns: 30
    perCodeBlockBasis: 1
    all:
      except:
        - myenv
```

**Listing 283: textwrap8.yaml**

```yaml
modifyLineBreaks:
  textWrapOptions:
    columns: 30
    perCodeBlockBasis: 1
    all:
      except:
        - masterDocument
```

**Listing 284: textwrap6.tex using Listing 281**

```latex
Before the environment; here
is a line of text that can be
wrapped by latexindent.pl.

\begin{myenv}
  Within the environment; here is a line of text that can be wrapped by latexindent.pl.
\end{myenv}

\begin{another}
  Within the environment; here is a line of text that can be wrapped by latexindent.pl.
\end{another}

After the environment; here
is a line of text that can be
wrapped by latexindent.pl.
```

**Listing 285: textwrap6.tex using Listing 282**

```latex
Before the environment; here
is a line of text that can be
wrapped by latexindent.pl.

\begin{myenv}
  Within the environment; here is a line of text that can be wrapped by latexindent.pl.
\end{myenv}

\begin{another}
  Within the environment; here is a line of text that can be wrapped by latexindent.pl.
\end{another}

After the environment; here
is a line of text that can be
wrapped by latexindent.pl.
```
6.1 Text Wrapping

<table>
<thead>
<tr>
<th>Listing 286: textwrap6.tex using Listing 283</th>
</tr>
</thead>
</table>

Before the environment; here is a line of text that can be wrapped by latexindent.pl.

\begin{myenv}
  Within the environment; here is a line of text that can be wrapped by latexindent.pl.
\end{myenv}

\begin{another}
  Within the environment; here is a line of text that can be wrapped by latexindent.pl.
\end{another}

After the environment; here is a line of text that can be wrapped by latexindent.pl.

Notice that:

- in Listing 284 the text wrapping routine has not been applied to any environments because it has been switched off (per-code-block) in Listing 281;
- in Listing 285 the text wrapping routine has not been applied to myenv because it has been switched off (per-name) in Listing 282;
- in Listing 286 the text wrapping routine has not been applied to masterDocument because of the settings in Listing 283.

The columns field has a variety of different ways that it can be specified; we've seen two basic ways already: the default (set to 0) and a positive integer (see Listing 279 on page 79, for example). We explore further options in Listings 287 to 289.

<table>
<thead>
<tr>
<th>Listing 287: textwrap9.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>modifyLineBreaks:</td>
</tr>
<tr>
<td>textWrapOptions:</td>
</tr>
<tr>
<td>columns:</td>
</tr>
<tr>
<td>default: 30</td>
</tr>
<tr>
<td>environments: 50</td>
</tr>
<tr>
<td>perCodeBlockBasis: 1</td>
</tr>
<tr>
<td>all: 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 288: textwrap10.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>modifyLineBreaks:</td>
</tr>
<tr>
<td>textWrapOptions:</td>
</tr>
<tr>
<td>columns:</td>
</tr>
<tr>
<td>default: 30</td>
</tr>
<tr>
<td>environments:</td>
</tr>
<tr>
<td>default: 50</td>
</tr>
<tr>
<td>perCodeBlockBasis: 1</td>
</tr>
<tr>
<td>all: 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 289: textwrap11.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>modifyLineBreaks:</td>
</tr>
<tr>
<td>textWrapOptions:</td>
</tr>
<tr>
<td>columns:</td>
</tr>
<tr>
<td>default: 30</td>
</tr>
<tr>
<td>environments:</td>
</tr>
<tr>
<td>myenv: 50</td>
</tr>
<tr>
<td>another: 15</td>
</tr>
<tr>
<td>perCodeBlockBasis: 1</td>
</tr>
<tr>
<td>all: 1</td>
</tr>
</tbody>
</table>

Listing 287 and Listing 288 are equivalent. Upon running the commands

```bash
$ latexindent.pl -s textwrap6.tex -l=textwrap9.yaml -m
$ latexindent.pl -s textwrap6.tex -l=textwrap11.yaml -m
```

we receive the respective output given in Listings 290 and 291.
6.1 Text Wrapping

**Listing 290: textwrap6.tex using Listing 287**

Before the environment; here is a line of text that can be wrapped by latexindent.pl.

\begin{myenv}
Within the environment; here is a line of text that can be wrapped by latexindent.pl.
\end{myenv}

\begin{another}
Within the environment; here is a line of text that can be wrapped by latexindent.pl.
\end{another}

After the environment; here is a line of text that can be wrapped by latexindent.pl.

**Listing 291: textwrap6.tex using Listing 289**

Before the environment; here is a line of text that can be wrapped by latexindent.pl.

\begin{myenv}
Within the environment; here is a line of text that can be wrapped by latexindent.pl.
\end{myenv}

\begin{another}
Within the environment; here is a line of text that can be wrapped by latexindent.pl.
\end{another}

After the environment; here is a line of text that can be wrapped by latexindent.pl.

Notice that:

- in Listing 290 the text for the masterDocument has been wrapped using 30 columns, while environments has been wrapped using 50 columns;

- in Listing 291 the text for myenv has been wrapped using 50 columns, the text for another has been wrapped using 15 columns, and masterDocument has been wrapped using 30 columns.

If you don't specify a default value on per-code-block basis, then the default value from columns will be inherited; if you don't specify a default value for columns then 80 will be used.

alignAtAmpersandTakesPriority is set to 1 by default; assuming that text wrapping is occurring on a per-code-block basis, and the current environment/code block is specified within Listing 33 on page 30 then text wrapping will be disabled for this code block.

If you wish to specify afterHeading commands (see Listing 122 on page 46) on a per-name basis, then you need to append the name with :heading, for example, you might use section:heading.
6.2 removeParagraphLineBreaks: modifying line breaks for paragraphs

When the \texttt{-m} switch is active \texttt{latexindent.pl} has the ability to remove line breaks from within paragraphs; the behaviour is controlled by the \texttt{removeParagraphLineBreaks} field, detailed in Listing 292. Thank you to [19] for shaping and assisting with the testing of this feature.

\begin{quote}
\textbf{removeParagraphLineBreaks: \langle fields \rangle}
\end{quote}

This feature is considered complimentary to the \texttt{oneSentencePerLine} feature described in Section 6.5 on page 92.

\begin{lstlisting}[language=yaml]
removeParagraphLineBreaks:
  all: 0
  beforeTextWrap: 0
  alignAtAmpersandTakesPriority: 1
  environments:
    quotation: 0
    ifElseFi: 0
    optionalArguments: 0
    mandatoryArguments: 0
    items: 0
    specialBeginEnd: 0
    afterHeading: 0
    preamble: 0
    filecontents: 0
    masterDocument: 0
\end{lstlisting}

This routine can be turned on \textit{globally} for every code block type known to \texttt{latexindent.pl} (see Table 1 on page 49) by using the \texttt{all} switch; by default, this switch is \texttt{off}. Assuming that the \texttt{all} switch is off, then the routine can be controlled on a per-code-block-type basis, and within that, on a per-name basis. We will consider examples of each of these in turn, but before we do, let's specify what \texttt{latexindent.pl} considers as a paragraph:

- it must begin on its own line with either an alphabetic or numeric character, and not with any of the code-block types detailed in Table 1 on page 49;
- it can include line breaks, but finishes when it meets either a blank line, a \texttt{\par} command, or any of the user-specified settings in the \texttt{paragraphsStopAt} field, detailed in Listing 309 on page 88.

Let's start with the \texttt{.tex} file in Listing 293, together with the YAML settings in Listing 294.

\begin{lstlisting}[language=tex]
\begin{myenv}
The lines in this environment are very short and contain many linebreaks.

Another paragraph.
\end{myenv}
\end{lstlisting}

\begin{lstlisting}[language=yaml]
modifyLineBreaks:
  removeParagraphLineBreaks:
    all: 1
\end{lstlisting}

Upon running the command

\texttt{cmh:~\$ latexindent.pl \texttt{-m shortlines.tex \texttt{-o shortlines1.tex \texttt{-l remove-para1.yaml}}}}
then we obtain the output given in Listing 295.

**Listing 295: shortlines1.tex**

```latex
\begin{myenv}
The lines in this environment are very short and contain many linebreaks.

Another paragraph.
\end{myenv}
```

Keen readers may notice that some trailing white space must be present in the file in Listing 293 which has crept in to the output in Listing 295. This can be fixed using the YAML file in Listing 418 on page 107 and running, for example,

```
cmh:~$ latexindent.pl -m shortlines.tex -o shortlines-tws.tex -l remove-para1.yaml,removeTWS-before.yaml
```

in which case the output is as in Listing 296; notice that the double spaces present in Listing 295 have been addressed.

**Listing 296: shortlines1-tws.tex**

```latex
\begin{myenv}
The lines in this environment are very short and contain many linebreaks.

Another paragraph.
\end{myenv}
```

Keeping with the settings in Listing 294, we note that the all switch applies to all code block types. So, for example, let’s consider the files in Listings 297 and 298

**Listing 297: shortlines-mand.tex**

```latex
\mycommand{The lines in this command are very short and contain many linebreaks.

Another paragraph.}
```

**Listing 298: shortlines-opt.tex**

```latex
\mycommand[The lines in this command are very short and contain many linebreaks.

Another paragraph.]
```

Upon running the commands

```
cmh:~$ latexindent.pl -m shortlines-mand.tex -o shortlines-mand1.tex -l remove-para1.yaml
cmh:~$ latexindent.pl -m shortlines-opt.tex -o shortlines-opt1.tex -l remove-para1.yaml
```

then we obtain the respective output given in Listings 299 and 300.

**Listing 299: shortlines-mand1.tex**

```latex
\mycommand{
The lines in this command are very short and contain many linebreaks.

Another paragraph.}
```

**Listing 300: shortlines-opt1.tex**

```latex
\mycommand[
The lines in this command are very short and contain many linebreaks.

Another paragraph.]
```
6.2 removeParagraphLineBreaks: modifying line breaks for paragraphs

\begin{one}
The lines in this environment are very short and contain many linebreaks.

Another paragraph.
\end{one}

\begin{two}
The lines in this environment are very short and contain many linebreaks.

Another paragraph.
\end{two}

Assuming that we turn off\textsuperscript{a} the all switch (by setting it to 0), then we can control the behaviour of removeParagraphLineBreaks either on a per-code-block-type basis, or on a per-name basis.

For example, let's use the code in Listing 301, and consider the settings in Listings 302 and 303; note that in Listing 302 we specify that every environment should receive treatment from the routine, while in Listing 303 we specify that only the one environment should receive the treatment.

\begin{listing}{301}
\mycommand
\begin{quote}
The lines in this command are very short and contain many linebreaks.

Another paragraph.
\end{quote}
\end{mycommand}

\end{listing}

Upon running the commands

\texttt{cmh:~$ latexindent.pl -m shortlines-envs.tex -o shortlines-envs2.tex -l remove-para2.yaml}
\texttt{cmh:~$ latexindent.pl -m shortlines-envs.tex -o shortlines-envs3.tex -l remove-para3.yaml}

then we obtain the respective output given in Listings 304 and 305.
The remaining code-block types can be customised in analogous ways, although note that commands, `keyEqualsValuesBracesBrackets`, `namedGroupingBracesBrackets`, `UnNamedGroupingBracesBrackets` are controlled by the `optionalArguments` and the `mandatoryArguments`.

The only special case is the `masterDocument` field; this is designed for 'chapter'-type files that may contain paragraphs that are not within any other code-blocks. For example, consider the file in Listing 306, with the YAML settings in Listing 307.
6.2 removeParagraphLineBreaks: modifying line breaks for paragraphs

The lines in this document are very short and contain many linebreaks.

Another paragraph.

\begin{myenv}
The lines in this document are very short and contain many linebreaks.
\end{myenv}

Listing 307: remove-para4.yaml

modifyLineBreaks:
  removeParagraphLineBreaks:
    masterDocument: 1

Upon running the following command

```
cmh:~$ latexindent.pl -m shortlines-md.tex -o shortlines-md4.tex -l remove-para4.yaml
```

then we obtain the output in Listing 308.

Listing 308: shortlines-md4.tex

The lines in this document are very short and contain many linebreaks.

Another paragraph.

\begin{myenv}
The lines in this document are very short and contain many linebreaks.
\end{myenv}

Note that the all field can take the same exceptions detailed in Listings 281 to 283.

The paragraph line break routine considers blank lines and the `\par` command to be the end of a paragraph; you can fine tune the behaviour of the routine further by using the paragraphsStopAt fields, shown in Listing 309.
The fields specified in paragraphsStopAt tell `latexindent.pl` to stop the current paragraph when it reaches a line that begins with any of the code-block types specified as 1 in Listing 309. By default, you’ll see that the paragraph line break routine will stop when it reaches an environment or verbatim code block at the beginning of a line. It is not possible to specify these fields on a per-name basis.

Let’s use the `.tex` file in Listing 310; we will, in turn, consider the settings in Listings 311 and 312.

Upon using the settings from Listing 307 on the previous page and running the commands

```
cmh:~$ latexindent.pl -m sl-stop.tex -o sl-stop4.tex -l remove-para4.yaml
```
```
cmh:~$ latexindent.pl -m sl-stop.tex -o sl-stop4-command.tex -l=remove-para4.yaml,stop-command.yaml
```
```
cmh:~$ latexindent.pl -m sl-stop.tex -o sl-stop4-comment.tex -l=remove-para4.yaml,stop-comment.yaml
```

we obtain the respective outputs in Listings 313 to 315; notice in particular that:

- in Listing 313 the paragraph line break routine has included commands and comments;
- in Listing 314 the paragraph line break routine has stopped at the `\texttt{emph}` command, because in Listing 311 we have specified commands to be 1, and `\texttt{emph}` is at the beginning of a line;
- in Listing 315 the paragraph line break routine has stopped at the comments, because in Listing 312 we have specified comments to be 1, and the comment is at the beginning of a line.

In all outputs in Listings 313 to 315 we notice that the paragraph line break routine has stopped at \begin{myenv} because, by default, environments is set to 1 in Listing 309.
6.3 Combining removeParagraphLineBreaks and textWrapOptions

The text wrapping routine (Section 6.1 on page 73) and remove paragraph line breaks routine (Section 6.2 on page 83) can be combined.

We motivate this feature with the code given in Listing 316.

Applying the text wrap routine from Section 6.1 on page 73 with, for example, Listing 275 on page 78 gives the output in Listing 317.

The text wrapping routine has behaved as expected, but it may be desired to remove paragraph line breaks before performing the text wrapping routine. The desired behaviour can be achieved by employing the beforeTextWrap switch.

Explicitly, using the settings in Listing 319 and running the command

```
cmh:~$ latexindent.pl -m textwrap7.tex -l=textwrap12.yaml -o=+-mod12
```

we obtain the output in Listing 318.
In Listing 318 the paragraph line breaks have first been removed from Listing 316, and then the text wrapping routine has been applied. It is envisaged that variants of Listing 319 will be among the most useful settings for these two features.

### 6.3.1 text wrapping beforeFindingChildCodeBlocks

I think it likely that most users will wish to employ the beforeFindingChildCodeBlocks option for the text wrap routine.

To motivate its use, we begin with the file in Listing 320.

Using the settings in Listing 319 and running

```bash
cmh:~$ latexindent.pl -m textwrap-bfccb.tex -l=textwrap12.yaml -o=+-mod12
```

gives the output in Listing 321

Note that we have added a 'ruler' to Listing 321 to assist with measuring.

The output in Listing 321 is not ideal, but it is expected. The reasoning is as follows:

- `latexindent.pl` first of all searches for code blocks (see Table 1 on page 49);
- it replaces each code block with a unique identifying string;
- with the settings of Listing 319 in place, it performs the paragraph line break removal, and then the text wrapping routine first of all on the `text` command, and then on the surrounding text;
- the surrounding text does not know that `text` is a command.

We can instruct `latexindent.pl` to perform text wrapping before searching for child code blocks by using the beforeFindingChildCodeBlocks field.
We save the quick-start settings from Listing 257 into Listing 322 and change the value of columns for demonstration. Upon running the command

```
cmh:~$ latexindent.pl -m textwrap-bfccb.tex -l=textwrap13.yaml -o=+-mod13
```

we receive the output in Listing 323.

**Listing 322: textwrap13.yaml (tweaked quick start)**

```yaml
modifyLineBreaks:
  textWrapOptions:
    columns: 40 #--- Changed from quick start
    perCodeBlockBasis: 1
    beforeFindingChildCodeBlocks: 1
    masterDocument: 1
    afterHeading: 1
    items: 1
  removeParagraphLineBreaks:
    masterDocument: 1
    afterHeading: 1
    items: 1
    beforeTextWrap: 1
```

**Listing 323: textwrap-bfccb-mod13.tex**

\begin{verbatim}
one two three four \text{(test five six
seven eight nine) ten eleven twelve
thirteen fourteen fifteen sixteen
seventeen}
\end{verbatim}

```
\begin{tabular}{|ccccccccccccccccc|}
  \hline
  \multicolumn{1}{|c}{0} & \multicolumn{1}{|c}{1} & \multicolumn{1}{|c}{2} & \multicolumn{1}{|c}{3} & \multicolumn{1}{|c}{4} & \multicolumn{1}{|c}{5} & \multicolumn{1}{|c}{6} & \multicolumn{1}{|c}{7} & \multicolumn{1}{|c}{8} & \multicolumn{1}{|c}{9} & \multicolumn{1}{|c}{10} & \multicolumn{1}{|c}{11} & \multicolumn{1}{|c}{12} & \multicolumn{1}{|c}{13} & \multicolumn{1}{|c|}{14} \\
\hline
  5 & 10 & 15 & 20 & 25 & 30 & 35 & 40 & 45 & 50 & 55 & 60 & 65 & 70 & 75 & 80 \\
\end{tabular}
```

This output is different from Listing 321, but is still not ideal, as the `test` command has indented its mandatory argument. We can employ `noAdditionalIndent` from Section 5.8 on page 48 in Listing 325 and run the command

```
cmh:~$ latexindent.pl -m textwrap-bfccb.tex -l=textwrap14.yaml -o=+-mod14
```

to receive the output in Listing 324.

**Listing 324: textwrap-bfccb-mod14.tex**

\begin{verbatim}
one two three four \text{(test five six
seven eight nine) ten eleven twelve
thirteen fourteen fifteen sixteen
seventeen}
\end{verbatim}

```
\begin{tabular}{|ccccccccccc|}
  \hline
  \multicolumn{1}{|c}{0} & \multicolumn{1}{|c}{1} & \multicolumn{1}{|c}{2} & \multicolumn{1}{|c}{3} & \multicolumn{1}{|c}{4} & \multicolumn{1}{|c}{5} & \multicolumn{1}{|c}{6} & \multicolumn{1}{|c}{7} & \multicolumn{1}{|c}{8} & \multicolumn{1}{|c|}{9} \\
\hline
  5 & 10 & 15 & 20 & 25 & 30 & 35 & 40 & 45 \\
\end{tabular}
```

**Listing 325: textwrap14.yaml**

```yaml
modifyLineBreaks:
  textWrapOptions:
    columns: 40
    perCodeBlockBasis: 1
    beforeFindingChildCodeBlocks: 1
    masterDocument: 1
    afterHeading: 1
    items: 1
  removeParagraphLineBreaks:
    masterDocument: 1
    afterHeading: 1
    items: 1
    beforeTextWrap: 1

noAdditionalIndent: #--- NEW BIT
test: 1 #--- NEW BIT
```

For reference, let's say that we had started from Listing 319, which instructs `latexindent.pl` to
apply the text-wrapping and paragraph-line-break-removal routines to all code blocks. In order to achieve the output in Listing 324, then we would need to employ an exception, which we demonstrate in Listing 326.

**Listing 326: textwrap15.yaml**

```yaml
modifyLineBreaks:
  textWrapOptions:
    columns: 40
    perCodeBlockBasis: 1
    beforeFindingChildCodeBlocks: 1
    all: 1
  removeParagraphLineBreaks:
    all:
      except:
      - test
    beforeTextWrap: 1

noAdditionalIndent:
  test: 1
```

### 6.4 Summary of text wrapping

I consider the most useful starting point for text wrapping to be given in Section 6.1.1 and Section 6.3.1.

Starting from Listing 257, it is likely that you will have to experiment with making adjustments (such as that given in Listing 325) depending on your preference.

It is important to note the following:

- verbatim code blocks of all types will **not** be affected by the text wrapping routine. See the demonstration in Listing 263 on page 75, together with environments: Listing 18 on page 27, commands: Listing 19 on page 27, noIndentBlock: Listing 20, specialBeginEnd: Listing 116 on page 45;
- comments will **not** be affected by the text wrapping routine (see Listing 265 on page 76);
- it is possible to wrap text on a per-code-block and a per-name basis;
- indentation is performed after the text wrapping routine; as such, indented code will likely exceed any maximum value set in the columns field.

### 6.5 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 [17] for helping to shape and test this feature. The behaviour of this part of the script is controlled by the switches detailed in Listing 327, all of which we discuss next.
6.5 oneSentencePerLine: modifying line breaks for sentences

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.

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.

For example, consider multiple-sentences.tex shown in Listing 328.

If we use the YAML files in Listings 330 and 332, and run the commands

```
cmh:~$ latexindent.pl multiple-sentences -m -i=manipulate-sentences.yaml
```

then we obtain the respective output given in Listings 329 and 331.
6.5 oneSentencePerLine: modifying line breaks for sentences

Notice, in particular, that the ‘internal’ sentence line breaks in Listing 328 have been removed in Listing 329, but have not been removed in Listing 331.

The remainder of the settings displayed in Listing 327 on the preceding page 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 333); 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 334); by default, this is only capital letters;
- end with a character (see Listing 335); 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.

### 6.5.1 sentencesFollow

Let’s explore a few of the switches in sentencesFollow; let’s start with Listing 328 on the previous page, and use the YAML settings given in Listing 337. Using the command

```bash
cmh:~$ latexindent.pl multiple-sentences -m -l=sentences-follow1.yaml
```
we obtain the output given in Listing 336.

**Listing 336: multiple-sentences.tex using Listing 337**

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.

We can explore the other field in Listing 333 with the .tex file detailed in Listing 338.

**Listing 338: 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

```
cmh:~$ latexindent.pl multiple-sentences1 -m -l=manipulate-sentences.yaml
```

then we obtain the respective output given in Listings 339 and 340.

**Listing 339: multiple-sentences1.tex using Listing 330 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.

**Listing 340: multiple-sentences1.tex using Listing 341**

(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 339 the first sentence after the ) has not been accounted for, but that following the inclusion of Listing 341, the output given in Listing 340 demonstrates that the sentence has been accounted for correctly.

### 6.5.2 sentencesBeginWith

By default, 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 334), and we can use the other field to define sentences to begin with other characters.
This is the first sentence.

\$a\$ can represent a number. 7 is at the beginning of this sentence.

Upon running the following commands

```
cmh:~$ latexindent.pl multiple-sentences2 -m -l=manipulate-sentences.yaml
cmh:~$ latexindent.pl multiple-sentences2 -m -l=manipulate-sentences.yaml,sentences-begin1.yaml
```

then we obtain the respective output given in Listings 343 and 344.

This is the first sentence.

\$a\$ can represent a number. 7 is at the beginning of this sentence.

Notice that in Listing 343, the first sentence has been accounted for but that the subsequent sentences have not. In Listing 344, all of the sentences have been accounted for, because the other field in Listing 345 has defined sentences to begin with either $ or any numeric digit, 0 to 9.

6.5.3 sentencesEndWith

Let’s return to Listing 328 on page 93; we have already seen the default way in which `latexindent.pl` will operate on the sentences in this file in Listing 329 on page 94. We can populate the other field with any character that we wish; for example, using the YAML specified in Listing 347 and the command

```
cmh:~$ latexindent.pl multiple-sentences -m -l=sentences-end1.yaml
cmh:~$ latexindent.pl multiple-sentences -m -l=sentences-end2.yaml
```

then we obtain the output in Listing 346.
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 346 and 348; in particular, in Listing 346 the word sentence has not been defined as a sentence, because we have not instructed latexindent.pl to begin sentences with lower case letters. We have changed this by using the settings in Listing 349, and the associated output in Listing 348 reflects this.

Referencing Listing 335 on page 94, 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.

Let’s consider the file shown in Listing 350.

Upon running the following commands

```
cmh:~$ latexindent.pl url -m -l=manipulate-sentences.yaml
```

we obtain the output given in Listing 351.

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 502 on page 127.

The basicFullStop routine should probably be avoided in most situations, as it does not accommodate the specifications above. For example, using the following command

```
cmh:~$ latexindent.pl url -m -l=alt-full-stop1.yaml
```

and the YAML in Listing 353 gives the output in Listing 352.
6.5 oneSentencePerLine: modifying line breaks for sentences

Notice that the full stop within the URL has not been accommodated correctly because of the non-default settings in Listing 353.

6.5.4 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.

For example, if we begin with the .tex file in Listing 354, and run the command

```bash
cmh:~$ latexindent.pl multiple-sentences3 -m -l=manipulate-sentences.yaml
```

then we obtain the output in Listing 355.

Furthermore, if sentences run across environments then, by default, the line breaks internal to the sentence will be removed. For example, if we use the .tex file in Listing 356 and run the commands

```bash
cmh:~$ latexindent.pl multiple-sentences4 -m -l=manipulate-sentences.yaml
cmh:~$ latexindent.pl multiple-sentences4 -m -l=keep-sen-line-breaks.yaml
```

then we obtain the output in Listings 357 and 358.
6.5 oneSentencePerLine: modifying line breaks for sentences

Once you've read Section 6.6, you will know that you can accommodate the removal of internal sentence line breaks by using the YAML in Listing 360 and the command

```shell
$ latexindent \multiple-sentences4 -m -l=item-rules2.yaml
```

the output of which is shown in Listing 359.

### 6.5.5 Text wrapping and indenting sentences

The oneSentencePerLine can be instructed to perform text wrapping and indentation upon sentences.

Let's use the code in Listing 361.

```latex
\begin{itemize}
  \item continues \end{itemize} across itemize and finishes here.
\end{itemize}
```

Referencing Listing 363, and running the following command

```shell
$ latexindent \multiple-sentences5 -m -l=sentence-wrap1.yaml
```
6.5 oneSentencePerLine: modifying line breaks for sentences

we receive the output given in Listing 362.

<table>
<thead>
<tr>
<th>Listing 362: multiple-sentences5.tex using Listing 363</th>
</tr>
</thead>
<tbody>
<tr>
<td>A distinção entre conteúdo \textit{real} e conteúdo \textit{intencional} está relacionada, ainda, à distinção entre o conceito husserliano de \textit{experiência} e o uso popular desse termo. No sentido comum, o \textbf{experimentado} é um complexo de eventos exteriores, e o \textbf{experimentar} consiste em percepções (além de julgamentos e outros atos) nas quais tais eventos aparecem como objetos, e objetos frequentemente relacionados ao ego empírico.</td>
</tr>
</tbody>
</table>

If you wish to specify the \texttt{columns} field on a per-code-block basis for sentences, then you would use \texttt{sentence}; explicitly, starting with Listing 287 on page 81, for example, you would replace/append environments with, for example, sentence: 50.

If you specify \texttt{textWrapSentences} as 1, but do not specify a value for \texttt{columns} then the text wrapping will \textit{not} operate on sentences, and you will see a warning in \texttt{indent.log}.

The indentation of sentences requires that sentences are stored as code blocks. This means that you may need to tweak Listing 335 on page 94. Let’s explore this in relation to Listing 364.

<table>
<thead>
<tr>
<th>Listing 364: multiple-sentences6.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consider the following: \begin{itemize} \item firstly. \item secondly. \end{itemize}</td>
</tr>
</tbody>
</table>

By default, \texttt{latexindent.pl} will find the full-stop within the first \texttt{item}, which means that, upon running the following commands

```
cmh:~$ latexindent.pl multiple-sentences6 -m -l=sentence-wrap1.yaml
```

we receive the respective output in Listing 365 and Listing 366.

<table>
<thead>
<tr>
<th>Listing 365: multiple-sentences6-mod1.tex using Listing 363</th>
</tr>
</thead>
<tbody>
<tr>
<td>Consider the following: \begin{itemize} \item firstly. \item secondly. \end{itemize}</td>
</tr>
</tbody>
</table>

We note that Listing 365 the \texttt{itemize} code block has \textit{not} been indented appropriately. This is because the oneSentencePerLine has been instructed to store sentences (because Listing 363); each sentence is then searched for code blocks.

We can tweak the settings in Listing 335 on page 94 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 367 (if you intend to
use this, ensure that you remove the line breaks from the other field).

Upon running

```
cmh:~$ latexindent.pl multiple-sentences6 -m -l=sentence-wrap1.yaml,itemize.yaml
```

we receive the output in Listing 368.

Consider the following:

```
\begin{itemize}
\item firstly.
\item secondly.
\end{itemize}
```

Notice that the sentence has received indentation, and that the \texttt{itemize} code block has been found and indented correctly.

6.6 Poly-switches

Every other field in the \texttt{modifyLineBreaks} field uses poly-switches, and can take one of the following integer values:

- 1 \textit{remove mode}: line breaks before or after the \texttt{<part of thing>} can be removed (assuming that \texttt{preserveBlankLines} is set to 0);

0 \textit{off mode}: line breaks will not be modified for the \texttt{<part of thing>} under consideration;

1 \textit{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>};

2 \textit{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>};

3 \textit{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;

4 \textit{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 \textit{begin statement}, \textit{body} or \textit{end statement} of the code blocks detailed in Table 1 on page 49. All poly-switches are \textit{off} by default; \texttt{latexindent.pl} searches first of all for \texttt{per-name} settings, and then followed by global \texttt{per-thing} settings.

6.6.1 Poly-switches for environments

We start by viewing a snippet of \texttt{defaultSettings.yaml} in Listing 369; note that it contains \textit{global} settings (immediately after the \texttt{environments} field) and that \texttt{per-name} settings are also allowed – in the case of Listing 369, settings for \texttt{equation*} have been specified for demonstration. Note that all poly-switches are \textit{off} (set to 0) by default.
6.6 Poly-switches

Let’s begin with the simple example given in Listing 370; note that we have annotated key parts of the file using ♠, ♥, ♦ and ♣, these will be related to fields specified in Listing 369.

Listing 370: env-mlb1.tex

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

6.6.1 Adding line breaks: BeginStartsOnOwnLine and BodyStartsOnOwnLine

Let’s explore BeginStartsOnOwnLine and BodyStartsOnOwnLine in Listings 371 and 372, and in particular, let’s allow each of them in turn to take a value of 1.

Listing 371: env-mlb1.yaml

```yaml
modifyLineBreaks:
environments:
  BeginStartsOnOwnLine: 1
```

Listing 372: env-mlb2.yaml

```yaml
modifyLineBreaks:
environments:
  BodyStartsOnOwnLine: 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 373 and 374 respectively.

Listing 373: env-mlb.tex using Listing 371

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

Listing 374: env-mlb.tex using Listing 372

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

There are a couple of points to note:

- in Listing 373 a line break has been added at the point denoted by ♠ in Listing 370; no other line breaks have been changed;
- in Listing 374 a line break has been added at the point denoted by ♥ in Listing 370; furthermore, note that the body of myenv has received the appropriate (default) indentation.

Let’s now change each of the 1 values in Listings 371 and 372 so that they are 2 and save them into env-mlb3.yaml and env-mlb4.yaml respectively (see Listings 375 and 376).

Listing 375: env-mlb3.yaml

```yaml
modifyLineBreaks:
environments:
  BeginStartsOnOwnLine: 2
```

Listing 376: env-mlb4.yaml

```yaml
modifyLineBreaks:
environments:
  BodyStartsOnOwnLine: 2
```

Upon running commands analogous to the above, we obtain Listings 377 and 378.

Listing 377: env-mlb.tex using Listing 375

```latex
% \begin{myenv}body of myenv\end{myenv}
```

Listing 378: env-mlb.tex using Listing 376

```latex
% \begin{myenv}body of myenv\end{myenv}
```
Note that line breaks have been added as in Listings 373 and 374, 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.

Let’s now change each of the 1 values in Listings 371 and 372 so that they are 3 and save them into `env-mlb5.yaml` and `env-mlb6.yaml` respectively (see Listings 379 and 380).

Upon running commands analogous to the above, we obtain Listings 381 and 382.

We will demonstrate this poly-switch value using the code in Listing 385.

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 386 and 387.

We note in particular that, by design, for this value of the poly-switches:

1. in Listing 386 a blank line has been inserted before the `\begin` statement, even though the `\begin` statement was already on its own line;
2. in Listing 387 a blank line has been inserted before the beginning of the body, even though it already began on its own line.

### 6.6.1.2 Adding line breaks using EndStartsOnOwnLine and EndFinishesWithLineBreak

Let's explore EndStartsOnOwnLine and EndFinishesWithLineBreak in Listings 388 and 389, and in particular, let's allow each of them in turn to take a value of 1.

After running the following commands:

```bash
$ latexindent -m env-mlb.tex -l env-mlb7.yaml
$ latexindent -m env-mlb.tex -l env-mlb8.yaml
```

the output is as in Listings 390 and 391.

There are a couple of points to note:

- in Listing 390 a line break has been added at the point denoted by ♦ in Listing 370 on page 102; no other line breaks have been changed and the \end{myenv} statement has not received indentation (as intended);
- in Listing 391 a line break has been added at the point denoted by ♣ in Listing 370 on page 102.

Let’s now change each of the 1 values in Listings 388 and 389 so that they are 2 and save them into env-mlb9.yaml and env-mlb10.yaml respectively (see Listings 392 and 393).

Upon running commands analogous to the above, we obtain Listings 394 and 395.

Note that line breaks have been added as in Listings 390 and 391, 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.

Let’s now change each of the 1 values in Listings 388 and 389 so that they are 3 and save them into env-mlb11.yaml and env-mlb12.yaml respectively (see Listings 396 and 397).

Upon running commands analogous to the above, we obtain Listings 398 and 399.
6.6 Poly-switches

Note that line breaks have been added as in Listings 390 and 391, and that a blank line has been added after the line break.

Let's now change each of the 1 values in Listings 396 and 397 so that they are 4 and save them into env-end4.yaml and env-end-f4.yaml respectively (see Listings 400 and 401).

We will demonstrate this poly-switch value using the code from Listing 385 on page 103.

Upon running the commands

```
cmh:~$ latexindent.pl -m env-mlb1.tex -l env-end4.yaml
```

then we receive the respective outputs in Listings 402 and 403.

We note in particular that, by design, for this value of the poly-switches:

1. in Listing 402 a blank line has been inserted before the `\end` statement, even though the `\end` statement was already on its own line;
2. in Listing 403 a blank line has been inserted after the `\end` statement, even though it already began on its own line.

### 6.6.1.3 poly-switches 1, 2, and 3 only add line breaks when necessary

If you ask `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. For example, if you process the file in Listing 404 using poly-switch values of 1, 2, or 3, it will be left unchanged.

Setting the poly-switches to a value of 4 instructs `latexindent.pl` to add a line break even if the `<part of thing>` is already on its own line; see Listings 386 and 387 and Listings 402 and 403.

In contrast, the output from processing the file in Listing 405 will vary depending on the poly-switches used; in Listing 406 you'll see that the comment symbol after the `\begin{myenv}` has been moved to the next line, as `BodyStartsOnOwnLine` is set to 1. In Listing 407 you'll see that the comment has been accounted for correctly because `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 405 and by setting the other poly-switches considered so far to 2 in turn.

The details of the discussion in this section have concerned global poly-switches in the environments field; each switch can also be specified on a per-name basis, which would take priority over the global values; with reference to Listing 369 on page 102, an example is shown for the equation* environment.

### 6.6.1.4 Removing line breaks (poly-switches set to −1)

Setting poly-switches to −1 tells `latexindent.pl` to remove line breaks of the `<part of the thing>`, if necessary. We will consider the example code given in Listing 408, noting in particular the positions of the line break highlighters, ♠, ♥, ♦ and ♣, together with the associated YAML files in Listings 409 to 412.

After running the commands

```
cmh:~$ latexindent.pl -m env-mlb4.tex -l env-mlb13.yaml
```
```
cmh:~$ latexindent.pl -m env-mlb4.tex -l env-mlb14.yaml
```
```
cmh:~$ latexindent.pl -m env-mlb4.tex -l env-mlb15.yaml
```
```
cmh:~$ latexindent.pl -m env-mlb4.tex -l env-mlb16.yaml
```

we obtain the respective output in Listings 413 to 416.
6.6 Poly-switches

Notice that in:

• Listing 413 the line break denoted by ♠ in Listing 408 has been removed;
• Listing 414 the line break denoted by ♥ in Listing 408 has been removed;
• Listing 415 the line break denoted by ♦ in Listing 408 has been removed;
• Listing 416 the line break denoted by♣ in Listing 408 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 409 to 412 into one file; alternatively, you could tell `latexindent.pl` to load them all by using the following command, for example:

```bash
$ latexindent.pl -m env-mlb5.tex -l env-mlb13.yaml,env-mlb14.yaml,env-mlb15.yaml,env-mlb16.yaml
```

which gives the output in Listing 370 on page 102.

### 6.6.1.5 About trailing horizontal space

Recall that on page 30 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; let's consider the file shown in Listing 417, which highlights trailing spaces.

The output from the following commands:

```bash
$ latexindent.pl -m env-mlb5.tex -l env-mlb13.yaml,env-mlb14.yaml,env-mlb15.yaml,env-mlb16.yaml
$ latexindent.pl -m env-mlb5.tex -l env-mlb13.yaml,env-mlb14.yaml,env-mlb15.yaml,env-mlb16.yaml,removeTWS-before.yaml
```

is shown, respectively, in Listings 419 and 420; note that the trailing horizontal white space has been preserved (by default) in Listing 419, while in Listing 420, it has been removed using the switch specified in Listing 418.

### 6.6.1.6 poly-switch line break removal and blank lines

Now let's consider the file in Listing 421, which contains blank lines.
6.6 Poly-switches

Listing 421: env-mlb6.tex

before words

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

after words

Upon running the following commands


we receive the respective outputs in Listings 423 and 424. In Listing 423 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 424, we have allowed the poly-switches to remove blank lines because, in Listing 422, we have set preserveBlankLines to 0.

Listing 422: UnpreserveBlankLines.yaml

modifyLineBreaks:
preserveBlankLines: 0

Listing 423: env-mlb6.tex using Listings 413 to 416

before words

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

after words

We can explore this further using the blank-line poly-switch value of 3; let’s use the file given in Listing 425.

Listing 425: env-mlb7.tex

\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

cmh:~$ latexindent.pl -m env-mlb7.tex -l env-mlb13.yaml,env-mlb14.yaml,UnpreserveBlankLines.yaml

we receive the outputs given in Listings 426 and 427.

Listing 426: env-mlb7-preserve.tex

\begin{one} one text \end{one} \begin{two} two text \end{two}

Listing 424: env-mlb6.tex using Listings 413 to 416 and Listing 422

before words\begin{myenv}body of myenv\end{myenv}after words

Listing 427: env-mlb7-preserve.tex

\begin{one} one text \end{one}

\begin{two} two text \end{two}
6.6 Poly-switches

Notice that in:

- Listing 426 that `\end{one}` has added a blank line, because of the value of `EndFinishesWithLineBreak` in Listing 397 on page 104, and even though the line break ahead of `\begin{two}` should have been removed (because of `BeginStartsOnOwnLine` in Listing 409 on page 106), the blank line has been preserved by default;

- Listing 427, by contrast, has had the additional line-break removed, because of the settings in Listing 422.

6.6.2 Poly-switches for double back slash

With reference to `lookForAlignDelims` (see Listing 33 on page 30) you can specify poly-switches to dictate the line-break behaviour of double back slashes in environments (Listing 35 on page 31), commands (Listing 69 on page 37), or special code blocks (Listing 104 on page 43). Note that for these poly-switches to take effect, the name of the code block must necessarily be specified within `lookForAlignDelims` (Listing 33 on page 30); we will demonstrate this in what follows.

Consider the code given in Listing 428.

Referencing Listing 428:

- DBS stands for *double back slash*;
- line breaks ahead of the double back slash are annotated by `\`, and are controlled by `DBSStartsOnOwnLine`;
- line breaks after the double back slash are annotated by `\`, and are controlled by `DBSFinishesWithLineBreak`.

Let's explore each of these in turn.

6.6.2.1 Double back slash starts on own line

We explore `DBSStartsOnOwnLine` (`\` in Listing 428); starting with the code in Listing 428, together with the YAML files given in Listing 430 and Listing 432 and running the following commands

```
cmh:~$ latexindent.pl -m tabular3.tex -l DBS1.yaml
```

then we receive the respective output given in Listing 429 and Listing 431.
6.6 Poly-switches

6.6.2 Double back slash finishes with line break

Let’s now explore DBSFinishesWithLineBreak (□ in Listing 428); starting with the code in Listing 428, together with the YAML files given in Listing 434 and Listing 436 and running the following commands

```
cmb:~$ latexindent.pl -m tabular3.tex -l DBS3.yaml
cmb:~$ latexindent.pl -m tabular3.tex -l DBS4.yaml
```

then we receive the respective output given in Listing 433 and Listing 435.

We note that

- Listing 434 specifies DBSFinishesWithLineBreak for every environment (that is within lookForAlignDelims, Listing 36 on page 31); the code following the double back slashes from Listing 428 has been moved to their own line in Listing 433;
- Listing 436 specifies DBSFinishesWithLineBreak on a per-name basis for tabular (that is within lookForAlignDelims, Listing 36 on page 31); the first double back slashes from Listing 428 have moved code following them to their own line in Listing 435, having added comment symbols before moving them; the final double back slashes have not added a line break as they are at the end of the body within the code block.

6.6.2.3 Double back slash poly-switches for specialBeginEnd

Let’s explore the double back slash poly-switches for code blocks within specialBeginEnd code blocks (Listing 102 on page 42); we begin with the code within Listing 437.
Upon using the YAML settings in Listing 439, and running the command
```
cmh:~$ latexindent.pl -m special4.tex -l DBS5.yaml
```
then we receive the output given in Listing 438.

There are a few things to note:

- in Listing 439 we have specified `cmhMath` within `lookForAlignDelims`; without this, the double back slash poly-switches would be ignored for this code block;
- the `DBSFinishesWithLineBreak` poly-switch has controlled the line breaks following the double back slashes;
- 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.

### 6.6.2.4 Double back slash poly-switches for optional and mandatory arguments

For clarity, we provide a demonstration of controlling the double back slash poly-switches for optional and mandatory arguments. We begin with the code in Listing 440.

```
\begin{mycommand}
1 & 2 & 3 \\
4 & 5 & 6 \\
\end{mycommand}
```

Upon using the YAML settings in Listings 442 and 444, 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 441 and 443.
6.6 Poly-switches

6.6.2.5 Double back slash optional square brackets

The pattern matching for the double back slash will also, optionally, allow trailing square brackets that contain a measurement of vertical spacing, for example `\*[3pt].

For example, beginning with the code in Listing 445

\begin{pmatrix}
1 & 2 \\ 3 & 4 \\
5 & 6 \\
7 & 8
\end{pmatrix}

and running the following command, using Listing 434,

```
~$ latexindent.pl -m pmatrix3.tex -l DBS3.yaml
```

then we receive the output given in Listing 446.

You can customise the pattern for the double back slash by exploring the `fine tuning` field detailed in Listing 502 on page 127.

6.6.3 Poly-switches for other code blocks

Rather than repeat the examples shown for the environment code blocks (in Section 6.6.1 on page 101), we choose to detail the poly-switches for all other code blocks in Table 2; 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 70 on page 37) can not be modified using `latexindent.pl`. However, there are two poly-switches available for `verbatim` code blocks: environments (Listing 18 on page 27), commands (Listing 19 on page 27) and `specialBeginEnd` (Listing 115 on page 45).
# 6.6 Poly-switches

<table>
<thead>
<tr>
<th>Code block</th>
<th>Sample</th>
<th>Poly-switch mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>environment</td>
<td><code>before words♠</code></td>
<td><code>♣ BeginStartsOnOwnLine</code></td>
</tr>
<tr>
<td></td>
<td>\begin{myenv}♥</td>
<td><code>♣ BodyStartsOnOwnLine</code></td>
</tr>
<tr>
<td></td>
<td><code>body of myenv♦</code></td>
<td><code>◇ EndStartsOnOwnLine</code></td>
</tr>
<tr>
<td></td>
<td><code>\end{myenv}♣</code></td>
<td><code>♣ EndFinishesWithLineBreak</code></td>
</tr>
<tr>
<td></td>
<td><code>after words</code></td>
<td></td>
</tr>
<tr>
<td>ifelsefi</td>
<td><code>before words♠</code></td>
<td><code>♣ IfStartsOnOwnLine</code></td>
</tr>
<tr>
<td></td>
<td>\if...♥</td>
<td><code>◇ BodyStartsOnOwnLine</code></td>
</tr>
<tr>
<td></td>
<td><code>body of if/or statement♠</code></td>
<td><code>▲ OrStartsOnOwnLine</code></td>
</tr>
<tr>
<td></td>
<td>\or♥</td>
<td><code>▼ OrFinishesWithLineBreak</code></td>
</tr>
<tr>
<td></td>
<td><code>body of if/or statement★</code></td>
<td><code>★ ElseStartsOnOwnLine</code></td>
</tr>
<tr>
<td></td>
<td>\else☺</td>
<td><code>□ ElseFinishesWithLineBreak</code></td>
</tr>
<tr>
<td></td>
<td><code>body of else statementdiamond</code></td>
<td><code>◇ FiStartsOnOwnLine</code></td>
</tr>
<tr>
<td></td>
<td><code>\fi♣</code></td>
<td><code>♣ FiFinishesWithLineBreak</code></td>
</tr>
<tr>
<td></td>
<td><code>after words</code></td>
<td></td>
</tr>
<tr>
<td>optionalArguments</td>
<td><code>...♠</code></td>
<td><code>♣ LSqBStartsOnOwnLine\(^8\)</code></td>
</tr>
<tr>
<td></td>
<td><code>[♥ value before comma★,</code></td>
<td><code>◇ OptArgBodyStartsOnOwnLine</code></td>
</tr>
<tr>
<td></td>
<td><code>□ end of body of opt argdiamond</code></td>
<td><code>★ CommaStartsOnOwnLine</code></td>
</tr>
<tr>
<td></td>
<td><code>]♠</code></td>
<td><code>□ CommaFinishesWithLineBreak</code></td>
</tr>
<tr>
<td></td>
<td><code>...</code></td>
<td><code>◇ RSqBStartsOnOwnLine</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>♣ RSqBFinishesWithLineBreak</code></td>
</tr>
<tr>
<td>mandatoryArguments</td>
<td><code>...♠</code></td>
<td><code>♣ LCuBStartsOnOwnLine\(^9\)</code></td>
</tr>
<tr>
<td></td>
<td><code>{♥ value before comma★,</code></td>
<td><code>◇ MandArgBodyStartsOnOwnLine</code></td>
</tr>
<tr>
<td></td>
<td><code>□ end of body of mand argdiamond</code></td>
<td><code>★ CommaStartsOnOwnLine</code></td>
</tr>
<tr>
<td></td>
<td><code>}</code>♠</td>
<td><code>□ CommaFinishesWithLineBreak</code></td>
</tr>
<tr>
<td></td>
<td><code>...</code></td>
<td><code>◇ RCuBStartsOnOwnLine</code></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>♣ RCuBFinishesWithLineBreak</code></td>
</tr>
<tr>
<td>commands</td>
<td><code>before words♠</code></td>
<td><code>♣ CommandStartsOnOwnLine</code></td>
</tr>
<tr>
<td></td>
<td><code>\mycommand♥</code></td>
<td><code>◇ CommandNameFinishesWithLineBreak</code></td>
</tr>
<tr>
<td></td>
<td><code>(arguments)</code></td>
<td></td>
</tr>
<tr>
<td>namedGroupingBraces Brackets</td>
<td><code>before words♠</code></td>
<td><code>♣ NameStartsOnOwnLine</code></td>
</tr>
<tr>
<td></td>
<td><code>\myname♥</code></td>
<td><code>◇ NameFinishesWithLineBreak</code></td>
</tr>
<tr>
<td></td>
<td><code>(braces/brackets)</code></td>
<td></td>
</tr>
<tr>
<td>keyEqualsValuesBracesBrackets</td>
<td><code>before words♠</code></td>
<td><code>♣ KeyStartsOnOwnLine</code></td>
</tr>
<tr>
<td></td>
<td><code>\key==♥</code></td>
<td><code>◇ EqualsStartsOnOwnLine</code></td>
</tr>
<tr>
<td></td>
<td><code>(braces/brackets)</code></td>
<td><code>◇ EqualsFinishesWithLineBreak</code></td>
</tr>
<tr>
<td>items</td>
<td><code>before words♠</code></td>
<td><code>♣ ItemStartsOnOwnLine</code></td>
</tr>
<tr>
<td></td>
<td><code>\item♥</code></td>
<td><code>◇ ItemFinishesWithLineBreak</code></td>
</tr>
<tr>
<td></td>
<td><code>...</code></td>
<td></td>
</tr>
<tr>
<td>specialBeginEnd</td>
<td><code>before words♠</code></td>
<td><code>♣ SpecialBeginStartsOnOwnLine</code></td>
</tr>
<tr>
<td></td>
<td><code>\[♥</code></td>
<td><code>◇ SpecialBodyStartsOnOwnLine</code></td>
</tr>
<tr>
<td></td>
<td><code>body of special/middle★</code></td>
<td><code>★ SpecialMiddleStartsOnOwnLine</code></td>
</tr>
<tr>
<td></td>
<td>\middle★</td>
<td><code>◇ SpecialMiddleFinishesWithLineBreak</code></td>
</tr>
<tr>
<td></td>
<td><code>body of special/middlediamond</code></td>
<td><code>◇ SpecialEndStartsOnOwnLine</code></td>
</tr>
<tr>
<td></td>
<td><code>\]♣</code></td>
<td><code>♣ SpecialEndFinishesWithLineBreak</code></td>
</tr>
<tr>
<td></td>
<td><code>after words</code></td>
<td></td>
</tr>
<tr>
<td>verbatim</td>
<td><code>before words♠</code></td>
<td><code>♣ VerbatimBeginStartsOnOwnLine</code></td>
</tr>
<tr>
<td></td>
<td><code>\begin{verbatim}♣</code></td>
<td></td>
</tr>
</tbody>
</table>

\(^8\) LSqB stands for Left Square Bracket  
\(^9\) LCuB stands for Left Curly Brace
### 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 2 on the preceding page) and **LCuBStartsOnOwnLine** for mandatory arguments, and **LSqBStartsOnOwnLine** for optional arguments.

Let's begin with the code in Listing 447 and the YAML settings in Listing 449; with reference to Table 2 on the previous page, the key **CommandNameFinishesWithLineBreak** is an alias for **BodyStartsOnOwnLine**.

**Listing 447: mycommand1.tex**

\mycommand
{ mand arg text
 mand arg text}
{ mand arg text
 mand arg text}

Upon running the command

```
$ latexindent -l=mycom-mlb1.yaml mycommand1.tex
```

we obtain Listing 448; note that the *second* mandatory argument beginning brace { has had its leading line break removed, but that the *first* brace has not.

**Listing 448: mycommand1.tex using Listing 449**

\mycommand
{ mand arg text
 mand arg text}{
 mand arg text
 mand arg text}

Now let's change the YAML file so that it is as in Listing 451; upon running the analogous command to that given above, we obtain Listing 450; both beginning braces { have had their leading line breaks removed.

**Listing 450: mycommand1.tex using Listing 451**

\mycommand{
 mand arg text
 mand arg text}{
 mand arg text
 mand arg text}

Now let's change the YAML file so that it is as in Listing 453; upon running the analogous command to that given above, we obtain Listing 452.
6.6 Poly-switches

6.6.5 Conflicting poly-switches: sequential code blocks

It is very easy to have conflicting poly-switches; if we use the example from Listing 447 on the preceding page, and consider the YAML settings given in Listing 455. The output from running

```
cmh:~$ latexindent.pl -m -l=mycom-mlb4.yaml mycommand1.tex
```

is given in Listing 455.

```
\mycommand
{  
  mand arg text  
  mand arg text} 
{  
  mand arg text  
  mand arg text}
```

Studying Listing 455, 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 454, 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.

We can explore this further by considering the YAML settings in Listing 457; upon running the command

```
cmh:~$ latexindent.pl -m -l=mycom-mlb5.yaml mycommand1.tex
```

we obtain the output given in Listing 456.

```
\mycommand
{  
  mand arg text  
  mand arg text} 
{  
  mand arg text  
  mand arg text}
```

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 459, which give associated output in Listing 458.
6.6 Poly-switches

LISTING 458: mycommand1.tex using Listing 459
\mycommand
{  mand arg text
    mand arg text}\
{  mand arg text
    mand arg text}

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.

6.6.6 Conflicting poly-switches: nested code blocks

Now let's consider an example when nested code blocks have conflicting poly-switches; we'll use the code in Listing 460, noting that it contains nested environments.

LISTING 460: nested-env.tex
\begin{one}
one text
\begin{two}
two text
\end{two}
\end{one}

Let's use the YAML settings given in Listing 462, which upon running the command

cmh:~$ latexindent.pl -m -l=nested-env-mlb1.yaml nested-env.tex

gives the output in Listing 461.

LISTING 461: nested-env.tex using Listing 462
\begin{one}
one text
\begin{two}
two text\end{two}\end{one}

In Listing 461, 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 latexindent.pl:

1. Phase 1: packing, in which code blocks are replaced with unique ids, working from the inside to the outside, and then sequentially – for example, in Listing 460, the two environment is found before the 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 −1);
   • line breaks at the end of the body can be added (if EndStartsOnOwnLine is 1 or 2) or removed (if EndStartsOnOwnLine is −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 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 NOT added (see EndFinishesWithLineBreak).

With reference to Listing 461, this means that during Phase 1:

• the two environment is found first, and the line break ahead of the \end{two} statement is removed because EndStartsOnOwnLine is set to –1. Importantly, because, at this stage, \end{two} does finish with a line break, EndFinishesWithLineBreak causes no action.

• next, the one environment is found; the line break ahead of \end{one} is removed because EndStartsOnOwnLine is set to −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 one environment will be found and processed first, followed by the two environment. If the two environment were to add a line break after the \end{two} statement, then latexindent.pl would have no way of knowing how much indentation to add to the subsequent text (in this case, \end{one}).

We can explore this further using the poly-switches in Listing 464; upon running the command

```
cmh:~$ latexindent.pl -m -l=nested-env-mlb2.yaml nested-env.tex
```

we obtain the output given in Listing 463.

---

During Phase 1:

• the two environment is found first, and the line break ahead of the \end{two} statement is not changed because EndStartsOnOwnLine is set to 1. Importantly, because, at this stage, \end{two} does finish with a line break, EndFinishesWithLineBreak causes no action.

• next, the one environment is found; the line break ahead of \end{one} is already present, and no action is needed.

The indentation is done in Phase 2, and then in Phase 3, the one environment is found and processed first, followed by the two environment. At this stage, the two environment finds EndFinishesWithLineBreak is −1, so it removes the trailing line break; remember, at this point, latexindent.pl has completely finished with the one environment.
SECTION 7

The -r, -rv and -rr switches

You can instruct \texttt{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 \textit{will} respect verbatim code blocks;
- the -rr switch will \textit{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 3.

<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 \texttt{replacements} field is shown in Listing 465; 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 465.

\begin{verbatim}
replacements:
  - amalgamate: 1
  -
    this: 'latexindent.pl'
    that: 'pl.latexindent'
  lookForThis: 1
  when: before
\end{verbatim}

The first entry within the \texttt{replacements} field is \texttt{amalgamate}, and is \textit{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 \textit{one} entry in the \texttt{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. With reference to Listing 465, the default action will replace every instance of the text \texttt{latexindent.pl} with \texttt{pl.latexindent}.

Beginning with the code in Listing 466 and running the command

\texttt{cmh:~$ latexindent.pl -r replace1.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

**Example 1** We begin with code given in Listing 470

```latex
\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.

Using the YAML in Listing 472, and running the command

```bash
$ latexindent.pl -r colsep.tex -l=colsep.yaml
```
then we achieve the output in Listing 471.

```
\begin{env}
  1 2 3
  4 5 6
\end{env}
```

Note that in Listing 472, 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 472 more concise by exploring the substitution field. Using the settings in Listing 474 and running the command

```
cmh:~$ latexindent.pl -r colsep.tex -l=colsep1.yaml
```

then we achieve the output in Listing 473.

```
\begin{env}
  1 2 3
  4 5 6
\end{env}
```

The code given in Listing 474 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, [10] for a detailed covering of the topic. With reference to Listing 474, 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 474 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 2** We’ll keep working with the file in Listing 470 on the preceding page for this example.

Using the YAML in Listing 476, and running the command

```
cmh:~$ latexindent.pl -r colsep.tex -l=multi-line.yaml
```

then we achieve the output in Listing 475.
With reference to Listing 476, we have specified a multi-line version of this by employing the literal YAML style \-r. See, for example, 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 465 on page 118. 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 478, and running the command

```
cmh:~$ latexindent.pl -r colsep.tex -l=multi-line1.yaml
```

then we achieve the output in Listing 477.

We note that, because we have specified \texttt{when: after}, that \texttt{latexindent.pl} has not found the string specified in Listing 478 within the file in Listing 470 on page 119. As it has looked for the string within Listing 478 \texttt{after} the indentation has been performed. After indentation, the string as written in Listing 478 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,

```
cmh:~$ 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 475.

**Example 3** An important part of the substitution routine is in \textit{capture groups}.

Assuming that we start with the code in Listing 479, let’s assume that our goal is to replace each occurrence of $$...$$ with \texttt{\begin{equation*}...\end{equation*}}. This example is partly motivated by \texttt{tex stackexchange question 242150}.
7.3 Examples of replacements

**LISTING 479: displaymath.tex**

before text $$a^2+b^2=4$$ and $$c^2$$

$$d^2+e^2 = f^2$$

and also $$g^2$$

$$$$ and some inline math: $$h^2$$

We use the settings in Listing 481 and run the command

```bash
$ latexindent.pl -r displaymath.tex -l=displaymath1.yaml
```

to receive the output given in Listing 480.

**LISTING 480: displaymath.tex using Listing 481**

before text \begin{equation*}a^2+b^2=4\end{equation*}

\begin{equation*}d^2+e^2 = f^2\end{equation*}

and also \begin{equation*}g^2\end{equation*}

\begin{equation*}\text{and some inline math: } h^2\end{equation*}

A few notes about Listing 481:

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 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 6.6 on page 101, which we do in Listing 483; upon running the command

```bash
$ latexindent.pl -r -m displaymath.tex -l=displaymath1.yaml,equation.yaml
```

then we receive the output in Listing 482.
### Example 4

This example is motivated by tex stackexchange question 490086. We begin with the code in Listing 484.

#### Listing 484: phrase.tex

<table>
<thead>
<tr>
<th>phrase 1</th>
<th>phrase 2</th>
<th>phrase 3</th>
<th>phrase 100</th>
</tr>
</thead>
<tbody>
<tr>
<td>phrase 1</td>
<td>phrase 2</td>
<td>phrase 3</td>
<td>phrase 100</td>
</tr>
<tr>
<td>phrase 1</td>
<td>phrase 2</td>
<td>phrase 3</td>
<td>phrase 100</td>
</tr>
<tr>
<td>phrase 1</td>
<td>phrase 2</td>
<td>phrase 3</td>
<td>phrase 100</td>
</tr>
</tbody>
</table>

Our goal is to make the spacing uniform between the phrases. To achieve this, we employ the settings in Listing 486, and run the command

```
cmh:~\$ latexindent -r phrase.tex -l=hspace.yaml
```

which gives the output in Listing 485.

#### Listing 485: phrase.tex using Listing 486

<table>
<thead>
<tr>
<th>phrase 1</th>
<th>phrase 2</th>
<th>phrase 3</th>
<th>phrase 100</th>
</tr>
</thead>
<tbody>
<tr>
<td>phrase 1</td>
<td>phrase 2</td>
<td>phrase 3</td>
<td>phrase 100</td>
</tr>
<tr>
<td>phrase 1</td>
<td>phrase 2</td>
<td>phrase 3</td>
<td>phrase 100</td>
</tr>
<tr>
<td>phrase 1</td>
<td>phrase 2</td>
<td>phrase 3</td>
<td>phrase 100</td>
</tr>
</tbody>
</table>

The \h+ setting in Listing 486 say to replace *at least one horizontal space* with a single space.
### Example 5

We begin with the code in Listing 487.

<table>
<thead>
<tr>
<th>Listing 487: references.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>equation \eqref{eq:aa} and Figure \ref{fig:bb} and table-\ref{tab:cc}</td>
</tr>
</tbody>
</table>

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 489 and running the command

```
cmh:~$ latexindent.pl -r references.tex -l=reference.yaml
```

which gives the output in Listing 488.

<table>
<thead>
<tr>
<th>Listing 488: references.tex using Listing 489</th>
</tr>
</thead>
<tbody>
<tr>
<td>\hyperref{equation \ref*{eq:aa}} and \hyperref{Figure \ref*{fig:bb}} and \hyperref{table \ref*{tab:cc}}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 489: reference.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>replacements:</td>
</tr>
<tr>
<td>substitution:</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

Referencing Listing 489, the | means or, we have used capture groups, together with an example of an optional pattern, (?:=eq)?.

### Example 6

Let's explore the three replacement mode switches (see Table 3 on page 118) in the context of an example that contains a verbatim code block, Listing 490; we will use the settings in Listing 491.

<table>
<thead>
<tr>
<th>Listing 490: verbi.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{myenv}</td>
</tr>
<tr>
<td>body of verbatim</td>
</tr>
<tr>
<td>\end{myenv}</td>
</tr>
<tr>
<td>some verbatim</td>
</tr>
<tr>
<td>\begin{verbatim}</td>
</tr>
<tr>
<td>body of verbatim text</td>
</tr>
<tr>
<td>\end{verbatim}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 491: verbatim1.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>replacements:</td>
</tr>
<tr>
<td>this: 'body'</td>
</tr>
<tr>
<td>that: 'head'</td>
</tr>
</tbody>
</table>

Upon running the following commands,
7.3 Examples of replacements

We note that:

1. in Listing 492 indentation has been performed, and that the replacements specified in Listing 491 have been performed, even within the verbatim code block;
2. in Listing 493 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 494 indentation has not been performed, but that replacements have been performed, not respecting the verbatim code block.

See the summary within Table 3 on page 118.

Example 7 Let’s explore the amalgamate field from Listing 465 on page 118 in the context of the file specified in Listing 495.

Let’s consider the YAML files given in Listings 496 to 498.

Upon running the following commands,

we receive the respective output in Listings 499 to 501.
7.3 Examples of replacements

We note that:

1. in Listing 499 the replacements from Listing 496 have been used;
2. in Listing 500 the replacements from Listings 496 and 497 have both been used, because the default value of `amalgamate` is 1;
3. in Listing 501 only the replacements from Listing 498 have been used, because the value of `amalgamate` has been set to 0.
SECTION 8

Fine tuning

latexindent.pl operates by looking for the code blocks detailed in Table 1 on page 49. The fine tuning of the details of such code blocks is controlled by the fineTuning field, detailed in Listing 502.

This field is for those that would like to peek under the bonnet/hood and make some fine tuning to latexindent.pl’s operating.

Making changes to the fine tuning may have significant consequences for your indentation scheme, proceed with caution!

The fields given in Listing 502 are all regular expressions. This manual is not intended to be a tutorial on regular expressions; you might like to read, for example, [10] for a detailed covering of the topic.

We make the following comments with reference to Listing 502:

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
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 72. In particular:
   (a) betterFullStop is in relation to the one sentence per line routine, detailed in Section 6.5 on page 92
   (b) doubleBackSlash is in relation to the DBSStartsOnOwnLine and DBSFinishesWithLineBreak polyswitches surrounding double back slashes, see Section 6.6.2 on page 109
   (c) comma is in relation to the CommaStartsOnOwnLine and CommaFinishesWithLineBreak polyswitches surrounding commas in optional and mandatory arguments; see Table 2 on page 113

It is not obvious from Listing 502, but each of the follow, before and between fields allow trailing comments, line breaks, and horizontal spaces between each character.

Example 8 As a demonstration, consider the file given in Listing 503, together with its default output using the command

```
cmh:~$ latexindent.pl finetuning1.tex
```

is given in Listing 504.

```latex
\mycommand{
 \rule{G -> +H[-G]CL}
 \rule{H -> -G[+H]CL}
 \rule{g -> +h[-g]cL}
 \rule{h -> -g[+h]cL}
 }
```

```latex
\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 504 that the indentation scheme has not worked as expected. We can fine tune the indentation scheme by employing the settings given in Listing 506 and running the command

```
cmh::$ latexindent.pl finetuning1.tex -l=fine-tuning1.yaml
```

and the associated (desired) output is given in Listing 505.

<table>
<thead>
<tr>
<th>Listing 505: finetuning1.tex using Listing 506</th>
</tr>
</thead>
<tbody>
<tr>
<td>\mycommand{</td>
</tr>
<tr>
<td>\rule{G -&gt; +H[-G]CL}</td>
</tr>
<tr>
<td>\rule{H -&gt; -G[+H]CL}</td>
</tr>
<tr>
<td>\rule{g -&gt; +h[-g]cL}</td>
</tr>
<tr>
<td>\rule{h -&gt; -g[+h]cL}</td>
</tr>
<tr>
<td>}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 506: finetuning1.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>fineTuning:</td>
</tr>
<tr>
<td>arguments:</td>
</tr>
<tr>
<td>between:</td>
</tr>
<tr>
<td>'_/&quot;*</td>
</tr>
</tbody>
</table>

Example 9 Let's have another demonstration; consider the file given in Listing 507, together with its default output using the command

```
cmh::$ latexindent.pl finetuning2.tex
```

is given in Listing 508.

<table>
<thead>
<tr>
<th>Listing 507: finetuning2.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>@misc{ wikilatex,</td>
</tr>
<tr>
<td>author = &quot;{Wikipedia contributors}&quot;&quot;,</td>
</tr>
<tr>
<td>title = &quot;LaTeX --- {Wikipedia}{,}&quot;&quot;,</td>
</tr>
<tr>
<td>note = &quot;[Online; accessed 3-March-2020]&quot;</td>
</tr>
<tr>
<td>}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 508: finetuning2.tex default</th>
</tr>
</thead>
<tbody>
<tr>
<td>@misc{ wikilatex,</td>
</tr>
<tr>
<td>author = &quot;{Wikipedia contributors}&quot;&quot;,</td>
</tr>
<tr>
<td>title = &quot;LaTeX --- {Wikipedia}{,}&quot;&quot;,</td>
</tr>
<tr>
<td>note = &quot;[Online; accessed 3-March-2020]&quot;</td>
</tr>
<tr>
<td>}</td>
</tr>
</tbody>
</table>

It's clear from Listing 508 that the indentation scheme has not worked as expected. We can fine tune the indentation scheme by employing the settings given in Listing 510 and running the command

```
cmh::$ latexindent.pl finetuning2.tex -l=fine-tuning2.yaml
```

and the associated (desired) output is given in Listing 509.

<table>
<thead>
<tr>
<th>Listing 509: finetuning2.tex using Listing 510</th>
</tr>
</thead>
<tbody>
<tr>
<td>@misc{ wikilatex,</td>
</tr>
<tr>
<td>author = &quot;{Wikipedia contributors}&quot;&quot;,</td>
</tr>
<tr>
<td>title = &quot;LaTeX --- {Wikipedia}{,}&quot;&quot;,</td>
</tr>
<tr>
<td>note = &quot;[Online; accessed 3-March-2020]&quot;</td>
</tr>
<tr>
<td>}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 510: finetuning2.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>fineTuning:</td>
</tr>
<tr>
<td>NamedGroupingBracesBrackets:</td>
</tr>
<tr>
<td>follow: '\h\R{[$}}{}'</td>
</tr>
<tr>
<td>UnNamedGroupingBracesBrackets:</td>
</tr>
<tr>
<td>follow: '{[&amp;}\{\}$}'</td>
</tr>
<tr>
<td>arguments:</td>
</tr>
<tr>
<td>between: '_/*</td>
</tr>
</tbody>
</table>

In particular, note that the settings in Listing 510 specify that NamedGroupingBracesBrackets and UnNamedGroupingBracesBrackets can follow " and that we allow --- between arguments.
Example 10  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 511 and running the following command:

\begin{verbatim}
cmh:\$ latexindent.p1 -m
-y='modifyLineBreaks:oneSentencePerLine:manipulateSentences:[:i-j,:i-k]
modifyLineBreaks:oneSentencePerLine:sentencesBeginWith:a-z:j,k
fineTuning:modifyLineBreaks:betterFullStop:j
"(?:\.\.|:!(?:[a-z]))(?:!:(!?:e\{g\}|(??:i\{e\}|(??:etc)))\.(?!([a-z][A-Z])
issue-243.tex -o=+-mod1
\end{verbatim}

\texttt{issues-243.tex -o=+-mod1}\texttt{issues-243.tex -o=+-mod1}

gives the output shown in Listing 512.

\begin{verbatim}
LISTING 511: finetuning3.tex
We go; you see: this sentence \cite{tex:stackexchange} finishes here.
\end{verbatim}

\begin{verbatim}
LISTING 512: finetuning3.tex using \texttt{-y} switch
We go; you see: this sentence \cite{tex:stackexchange} finishes here.
\end{verbatim}

Example 11  We can tweak the fineTuning for how trailing comments are classified. For motivation, let’s consider the code given in Listing 513.

\begin{verbatim}
LISTING 513: finetuning4.tex
some before text \href{Handbook\%20for\%30Spoken\%40document.pdf}{my document}
some after text
\end{verbatim}

We will compare the settings given in Listings 514 and 515.

\begin{verbatim}
LISTING 514: href1.yaml
modifyLineBreaks:
  textWrapOptions:
    columns: 80
    all: 1
    perCodeBlockBasis: 1
removeParagraphLineBreaks:
  all: 1

LISTING 515: href2.yaml
modifyLineBreaks:
  textWrapOptions:
    columns: 80
    all: 1
    perCodeBlockBasis: 1
removeParagraphLineBreaks:
  all: 1
fineTuning:
  trailingComments:
    notPreceededBy:
      '(?:(?<!Handbook)(?!for)(?!Spoken))'

Upon running the following commands
\begin{verbatim}
cmh:\$ latexindent.p1 -m finetuning4.tex -o=+-mod1 -l=href1
cmh:\$ latexindent.p1 -m finetuning4.tex -o=+-mod2 -l=href2
\end{verbatim}

we receive the respective output in Listings 516 and 517.

\begin{verbatim}
LISTING 516: finetuning4.tex using Listing 514
some before text \href{Handbook some after text\%20for\%30Spoken\%40document.pdf}{my document}
\end{verbatim}
We note that in:

- Listing 516 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 517 has fine-tuned the trailing comment matching, and says that \% cannot be immediately preceeded 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.

Another approach to this situation, which does not use fineTuning, is to use noIndentBlock which we discussed in Listing 20 on page 27; using the settings in Listing 518 and running the command

```
cmh:~\$ latexindent.pl -m finetuning4.tex -o=-mod3 -l:href3
```

then we receive the same output given in Listing 517; see also paragraphsStopAt in Listing 309 on page 88.

**Listing 518: href3.yaml**

```
modifyLineBreaks:
  textWrapOptions:
    columns: 80
    all: 1
    perCodeBlockBasis: 1
  removeParagraphLineBreaks:
    all: 1
    paragraphsStopAt:
      verbatim: 0
  noIndentBlock:
    href:
      begin: '\\href\{[^}\]*?\}\{
      body: '[^}\]*\'
      end: '\}'
```

With reference to the body field in Listing 518, 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.
Conclusions and known limitations

There are a number of known limitations of the script, and almost certainly quite a few that are unknown!

For example, with reference to the multicolumn alignment routine in Listing 46 on page 33, when working with code blocks in which multicolumn commands overlap, the algorithm can fail.

Another limitation is to do with 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 116); 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 \texttt{fileExtensionPreference} (see Listing 16 on page 25) will be consulted. If you find a case in which the script struggles, please feel free to report it at \url{[11]}, and in the meantime, consider using a \texttt{noIndentBlock} (see page 27).

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 \url{[11]}; otherwise, feel free to find me on the \url{http://tex.stackexchange.com/users/6621/cmhughes}. 
10.1 External links

[21] perldoc Encode::Supported. URL: https://perldoc.perl.org/Encode::Supported (visited on 05/06/2021).
[27] Video demonstration of latexindent.pl on youtube. URL: https://www.youtube.com/watch?v=wo38aah2F4E&spfreload=10 (visited on 02/21/2017).

10.2 Contributors


SECTION A

Required Perl modules

If you intend to use `latexindent.pl` and not one of the supplied standalone executable files, then you will need a few standard Perl modules – if you can run the minimum code in Listing 519 (`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 519: helloworld.pl**

```perl
#!/usr/bin/perl
use strict;
use warnings;
use utf8;
use PerlIO::encoding;
use Unicode::GCString;
use open ':std', ':encoding(UTF-8)';
use Text::Wrap;
use Text::Tabs;
use FindBin;
use YAML::Tiny;
use File::Copy;
use File::Basename;
use File::HomeDir;
use Encode;
use Getopt::Long;
use Data::Dumper;
use List::Util qw(max);
print "hello\_world";
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 installed modules

Manually installing the modules given in Listing 519 will vary depending on your operating system and Perl distribution.
A.2.1 Linux

Linux users may be interested in exploring Perlbrew [20]; an example installation would be:

```bash
cmh:~$ sudo apt-get install perlbrew
cmh:~$ perlbrew init
cmh:~$ perlbrew install perl-5.28.1
cmh:~$ perlbrew switch perl-5.28.1
cmh:~$ sudo apt-get install curl
cmh:~$ curl -L http://cpanmin.us | perl -e 'cpanminus

For other distributions, the Ubuntu/Debian approach may work as follows

```bash
cmh:~$ sudo apt install perl
cmh:~$ sudo cpan -i App::cpanminus
cmh:~$ sudo cpanm YAML::Tiny
cmh:~$ sudo cpanm File::HomeDir
cmh:~$ sudo cpanm Unicode::GCString
```

or else by running, for example,

```bash
cmh:~$ sudo perl -MCPAN -e 'install "File::HomeDir"

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 520; thanks to [12] for providing these details.

```bash
# Listing 520: 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
  cpanm -n Unicode::GCString
```

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:
A.2 Manually installed modules

```bash
cmh:~$ brew install perl
cmh:~$ brew install cpanm
cmh:~$
cmh:~$ cpanm YAML::Tiny
cmh:~$ cpanm File::HomeDir
cmh:~$ cpanm Unicode::GCString
```

A.2.3 Windows

Strawberry Perl users on Windows might use CPAN client. All of the modules are readily available on CPAN [5].

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

```bash
C:\Users\cmh> latexindent.exe -t myfile.tex
```
latexindent.pl has a few scripts (available at [11]) that can update the path variables. Thank you to [14] for this feature. If you're on a Linux or Mac machine, then you'll want CMakeLists.txt from [11].

### 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 [11];
2. within your directory, create a directory called path-helper-files and download CMakeLists.txt and cmake_uninstall.cmake.in from [11]/path-helper-files to this directory;
3. run

   ```bash
   cmh:~$ ls /usr/local/bin
   ```

   to see what is currently in there;
4. run the following commands

   ```bash
   cmh:~$ sudo apt-get install cmake
cmh:~$ sudo apt-get update && sudo apt-get install build-essential
cmh:~$ mkdir build && cd build
cmh:~$ cmake ../path-helper-files
cmh:~$ sudo make install
   ```

5. run

   ```bash
   cmh:~$ ls /usr/local/bin
   ```

   again to check that latexindent.pl, its modules and defaultSettings.yaml have been added.

To remove the files, run

```bash
cmh:~$ 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 [11] to your chosen directory;
2. open a command prompt and run the following command to see what is currently in your %path% variable;
B.2 Add to path for Windows

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

logFilePreferences

Listing 17 on page 26 describes the options for customising the information given to the log file, and we provide a few demonstrations here. Let's say that we start with the code given in Listing 521, and the settings specified in Listing 522.

**Listing 521: simple.tex**
```latex
\begin{myenv}
body of myenv
\end{myenv}
```

**Listing 522: logfile-prefs1.yaml**
```yaml
logFilePreferences:
  showDecorationStartCodeBlockTrace: "+++++
  showDecorationFinishCodeBlockTrace: "-----"
```

If we run the following command (noting that `-t` is active)
```
cmh:~$ latexindent.pl -t -l=logfile-prefs1.yaml simple.tex
```
then on inspection of `indent.log` we will find the snippet given in Listing 523.

**Listing 523: indent.log**
```
+++++
TRACE: environment found: myenv
No ancestors found for myenv
Storing settings for myenv environments
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.
SECTION D

Encoding indentconfig.yaml

In relation to Section 4 on page 21, 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 524 within their `indentconfig.yaml`, where 936 is the result of the `chcp` command.

```
Listing 524: encoding demonstration for indentconfig.yaml

encoding: cp936
```
SECTION E

dos2unix linebreak adjustment

dos2unixlinebreaks: (integer)

If you use 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 dos2unixlinebreaks to 1 and employing, for example, the following command.

cmb:~$ latexindent.pl -y="dos2unixlinebreaks:1" myfile.tex

See [29] for further details.
SECTION F

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,

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

whereas in Version 3.0 you would run any of the following, for example,

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

noting that the *output* file is given next to the `-o` switch.

The fields given in Listing 525 are *obsolete* from Version 3.0 onwards.

<table>
<thead>
<tr>
<th>Listing 525: 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 526 and 527

<table>
<thead>
<tr>
<th>Listing 526: 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 527: 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 528; as of Version 3.0, you would write YAML as in Listing 529 or, if you're using `-m` switch, Listing 530.
LISTING 528: noAdditionalIndent in Version 2.2

noAdditionalIndent:
\[: 0
\]: 0

LISTING 529: noAdditionalIndent for displayMath in Version 3.0

specialBeginEnd:
displayMath:
  begin: '\\['
  end: '\\']'
lookForThis: 0

LISTING 530: noAdditionalIndent for displayMath in Version 3.0

noAdditionalIndent:
displayMath: 1

End
Index

--- B ---
backup files
  cycle through, 26
  extension settings, 25
  maximum number of backup files, 26
  number of backup files, 26
  overwrite switch, -w, 14

--- C ---
capturing parenthesis (regex), 39

--- D ---
delimiters, 110
  advanced settings, 31
  advanced settings of lookForAlignDelims, 30
  ampersand &, 31
  delimiter justification (left or right), 39
  delimiterRegEx, 39
  dontMeasure feature, 37
  double backslash demonstration, 36
  lookForAlignDelims, 31
  poly-switches for double back slash, 109
  spacing demonstration, 32
  with specialBeginEnd and the -m switch, 111
  within specialBeginEnd blocks, 45

--- I ---
indentation
  customising indentation per-code block, 48
  customising per-name, 48
  default, 17
  defaultIndent description, 30
  defaultIndent using -y switch, 17
  defaultIndent using YAML file, 21
  maximum indetation, 47
  no additional indent, 48
  no additional indent global, 48
  removing indentation per-code block, 48
  summary, 66

--- L ---
linebreaks
  summary of poly-switches, 110

--- M ---
modifying linebreaks
  at the beginning of a code block, 102
  at the end of a code block, 104
  by text wrapping, globally, 74
  by text wrapping, per-code-block, 77
  by using one sentence per line, 92
  surrounding double back slash, 109
  using poly-switches, 101

--- P ---
poly-switches
  adding blank lines (again!): set to 4, 103, 105
  adding blank lines: set to 3, 103, 104
  adding comments and then line breaks: set to 2, 102, 104
  adding line breaks: set to 1, 102, 104
  blank lines, 107
  conflicting partnering, 114
  conflicting switches, 115, 116
  default values, 101
  definition, 101
  double backslash, 110
  environment global example, 101
  environment per-code block example, 101
  for double back slash (delimiters), 109–112
  off by default: set to 0, 101
  removing line breaks: set to -1, 106
  summary of all poly-switches, 112
  values, 101
  visualisation: ♠, ◐, ♦, ♣, 102

--- R ---
regular expressions
  a word about, 12
  ampersand alignment, 31
  arguments, 127
  at least one +, 46, 120, 123, 127–129
  capturing parenthesis, 39
  character class demonstration, 131
  commands, 127
  delimiter alignment for edge or node, 45
  delimiter regex at #, 41
  delimiter regex at # or >, 41
  delimiter regex at = or >, 40
  delimiter regex at only >, 40
  delimiterRegEx, 31
  dontMeasure feature, cell, 38
  dontMeasure feature, row, 39
  environments, 127
  fine tuning, 127
  horizontal space \h, 46, 48, 101, 123, 124, 127
  ifElseFi, 127
  keyEqualsValuesBracesBrackets, 127
  lowercase alph a-z, 38, 39, 48, 92, 94, 96, 101, 127
modifyLineBreaks, 127
NamedGroupingBracesBrackets, 127
numeric 0-9, 45, 48, 96, 101, 127
replacement switch, -r, 119
substitution field, arraycolsep, 120
substitution field, equation, 122
UnNamedGroupingBracesBrackets, 127
uppercase alph A-Z, 45, 48, 92, 94, 101, 127
using -y switch, 23

— S —
sentences
begin with, 94, 95
end with, 94, 96
follow, 94
indenting, 99
one sentence per line, 92
oneSentencePerLine, 92
removing sentence line breaks, 93
text wrapping, 99

specialBeginEnd
alignment at delimiter, 45
combined with lookForAlignDelims, 45
default settings, 42
delimiterRegEx, 45
delimiterRegEx tweaked, 46
double backslash poly-switch demonstration, 110
IfElsFi example, 44
indentRules example, 61
indentRulesGlobal, 66
introduction, 42
lookForAlignDelims, 110
middle, 44
noAdditionalIndent, 61
noAdditionalIndentGlobal, 66
paragraphsStopAt, 87
poly-switch summary, 112
removeParagraphLineBreaks, 83
searching for special before commands, 43
specifying as verbatim, 45
textWrapOptions, 77
tikz example, 45
update to displaymath V3.0, 143

switches
-c, --cruft definition and details, 18
-d, --onlydefault definition and details, 18
-g, --logfile definition and details, 18
-h, --help definition and details, 14
-l in relation to other settings, 23
-l, --local definition and details, 16
-m demonstration, 73–81, 83–85, 87–89, 93–112, 114–117, 122
-m, --modifylinebreaks definition and details, 18
-o demonstration, 36, 41, 45, 74–77, 83–85, 87–89, 124, 143
-o, --output definition and details, 15
-r demonstration, 118–125
-r, --replacement definition and details, 19
-rr demonstration, 121, 124
-rr, --onlyreplacement definition and details, 19
-rv demonstration, 124
-rv, --replacementrespectverb definition and details, 19
-s, --silent definition and details, 16
-sl, --screenlog definition and details, 18
-t, --trace definition and details, 16
-tp, --ttrace definition and details, 16
-v, --version definition and details, 14
-w, --overwrite definition and details, 14
-y demonstration, 23, 36, 100
-yc, --yaml definition and details, 17

— T —
text wrap
quick start, 74
recommended starting point, 90

— V —
verbatim
commands, 27
comparison with -r and -rr switches, 124
environments, 27
in relation to oneSentencePerLine, 98
in relation to paragraphsStopAt, 87
in relation to textWrapOptions, 75
noIndentBlock, 27
poly-switch summary, 112
rv, replacementrespectverb switch, 19, 118
specialBeginEnd, 45
verbatimEnvironments demonstration (-l switch), 23
verbatimEnvironments demonstration (-y switch), 23
within summary of text wrapping, 92

— W —
warning
amalgamate field, 71
be sure to test before use, 10
capturing parenthesis for lookForAlignDelims, 39
changing huge (textwrap), 76
editing YAML files, 22
fine tuning, 127
the m switch, 73