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 10.

Contents

1 Introduction 9
  1.1 Thanks ................................................................. 9
  1.2 License ............................................................... 9
  1.3 About this documentation ........................................... 9
  1.4 Quick start ........................................................... 10
  1.5 A word about regular expressions .................................. 11

2 Demonstration: before and after 12

3 How to use the script 13
  3.1 From the command line ............................................... 13
  3.2 From arara ............................................................. 19

4 indentconfig.yaml, local settings and the -y switch 20
  4.1 indentconfig.yaml and .indentconfig.yaml ....................... 20
  4.2 localSettings.yaml .................................................. 21
  4.3 The -y|yaml switch .................................................. 21
  4.4 Settings load order ................................................ 22

5 defaultSettings.yaml 24
  5.1 lookForAlignDelims: the dontMeasure feature .................. 33

* and contributors! See Section 10.2 on page 122. For all communication, please visit [8].
A.2 Manually installed modules .............................................. 123

B Updating the path variable .................................................. 125
   B.1 Add to path for Linux .................................................. 125
   B.2 Add to path for Windows .............................................. 125

C logFilePreferences .............................................................. 127

D Differences from Version 2.2 to 3.0 ........................................ 128

---

Listings

LISTING 1: demo-tex.tex ......................................................... 9
LISTING 2: fileExtensionPreference ......................................... 10
LISTING 3: modifyLineBreaks ............................................... 10
LISTING 4: replacements .................................................... 10
LISTING 5: Possible error messages ....................................... 11
LISTING 6: filecontents1.tex ............................................... 12
LISTING 7: filecontents1.tex default output ........................... 12
LISTING 8: tikzset.tex ........................................................ 12
LISTING 9: tikzset.tex default output ................................... 12
LISTING 10: pstricks.tex ..................................................... 12
LISTING 11: pstricks.tex default output ................................ 12
LISTING 15: fileExtensionPreference .................................... 24
LISTING 16: logFilePreferences ........................................... 25
LISTING 17: verbatimEnvironments ....................................... 26
LISTING 18: verbatimCommands ........................................... 26
LISTING 19: noIndentBlock .................................................. 26
LISTING 20: noIndentBlock demonstration .............................. 26
LISTING 21: removeTrailingWhitespace ................................. 26
LISTING 23: fileContentsEnvironments .................................. 27
LISTING 24: lookForPreamble ............................................... 27
LISTING 25: Motivating preambleCommandsBeforeEnvironments .. 27
LISTING 26: default output .................................................. 27
LISTING 27: tabular1.tex ..................................................... 28
LISTING 28: tabular1.tex default output ............................... 28
LISTING 29: lookForAlignDelims (advanced) ............................ 28
LISTING 30: tabular2.tex ..................................................... 29
LISTING 31: tabular2.yaml ................................................... 29
LISTING 32: tabular3.yaml ................................................... 29
LISTING 33: tabular4.yaml ................................................... 30
LISTING 34: tabular5.yaml ................................................... 30
LISTING 35: tabular6.yaml ................................................... 30
LISTING 36: tabular7.yaml ................................................... 30
LISTING 37: tabular8.yaml ................................................... 30
LISTING 38: tabular2.tex default output ................................ 30
LISTING 39: tabular2.tex using Listing 31 ............................ 30
LISTING 40: tabular2.tex using Listing 32 ............................ 30
LISTING 41: tabular2.tex using Listings 31 and 33 ................... 31
LISTING 42: tabular2.tex using Listings 31 and 34 ................... 31
LISTING 43: tabular2.tex using Listings 31 and 35 ................... 31
LISTING 44: tabular2.tex using Listings 31 and 36 ................... 31
LISTING 45: tabular2.tex using Listings 31 and 37 ................... 32
LISTING 46: tabular4.tex ..................................................... 32
LISTING 47: tabular4.default.tex .......................................... 32
LISTING 48: tabular4-FDBS.tex ............................................ 32
LISTING 49: matrix1.tex ..................................................... 33
LISTING 50: matrix1.tex default output ................................ 33
LISTING 51: align-block.tex ............................................... 33
LISTING 52: align-block.tex default output ............................ 33
LISTING 53: tabular-DM.tex ................................................ 33
LISTING 54: tabular-DM.tex default output ............................. 33
LISTING 55: tabular-DM.tex using Listing 56 .......................... 34
LISTING 56: dontMeasure1.yaml .......................................... 34
LISTING 57: tabular-DM.tex using Listing 58 or Listing 60 ........ 34
LISTING 58: dontMeasure2.yaml .......................................... 34
LISTING 59: tabular-DM.tex using Listing 60 or Listing 60 ........ 34
LISTING 60: dontMeasure3.yaml .......................................... 34
LISTING 61: dontMeasure4.yaml .......................................... 34
LISTING 62: tabular-DM.tex using Listing 63 .......................... 35
LISTING 63: dontMeasure5.yaml .......................................... 35
LISTING 64: tabular-DM.tex using Listing 65 .......................... 35
LISTING 65: dontMeasure6.yaml .......................................... 35

[git] • master @ fc35190 • 2020-03-21 • V3.8
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 \texttt{\LaTeX} stack exchange [1] I received some positive feedback and follow-up feature requests. A big thank you to Harish Kumar [10] 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 122 for their valued contributions, and thank you to those who report bugs and request features at [8].

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 24) 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:

\begin{quotation}
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.
\end{quotation}

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 [8] with a complete minimum working example as I would like to improve the code as much as possible.

\begin{caution}
Before you try the script on anything important (like your thesis), test it out on the sample files in the test-case directory [8].
\end{caution}

If you have used any version 2.* of \texttt{latexindent.pl}, there are a few changes to the interface; see appendix D on page 128 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 490. 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{verbatim}
LISTING 1: demo-tex.tex
demonstration .tex file
\end{verbatim}

This type of listing is a .\texttt{tex} file.
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.

This type of listing is a .yaml file; when you see line numbers given (as here) it means that the snippet is taken directly from defaultSettings.yaml, discussed in detail in Section 5 on page 24.

This type of listing is a .yaml file, but it will only be relevant when the \texttt{-m} switch is active; see Section 6 on page 66 for more details.

This type of listing is a .yaml file, but it will only be relevant when the \texttt{-r} switch is active; see Section 7 on page 109 for more details.

You will occasionally see dates shown in the margin (for example, next to this paragraph!) which detail the date of the version in which the feature was implemented; the 'N' stands for 'new as of the date shown' and 'U' stands for 'updated as of the date shown'. If you see \textasteriskcentered N, it means that the feature is either new (N) or updated (U) as of the release of the current version; if you see \textasteriskcentered U attached to a listing, then it means that listing is new (N) or updated (U) as of the current version. If you have not read this document before (and even if you have!), then you can ignore every occurrence of the \textasteriskcentered N; they are simply there to highlight new and updated features. The new and updated features in this documentation (V3.8) are on the following pages:

- align final double back slash (N) ..................................................... 29
- don't measure feature (N) ................................................................. 29
- delimiter RegEx feature (N) ............................................................... 29
- delimiter justification (N) ............................................................... 29
- alignFinalDoubleBackslash demonstration (N) ................................. 32
- don't measure feature (N) ................................................................. 33
- delimiterRegEx feature (N) ............................................................... 35

N: 2017-06-25
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 [7].

### Listing 5: Possible error messages

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

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

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

You might also like to see [https://stackoverflow.com/questions/19590042/error-cant-locate-file-homedir-pm-in-inc](https://stackoverflow.com/questions/19590042/error-cant-locate-file-homedir-pm-in-inc), for example, as well as appendix A on page 123.
SECTION 2

Demonstration: before and after

Let’s give a demonstration of some before and after code – after all, you probably won’t want to try the script if you don’t much like the results. You might also like to watch the video demonstration I made on youtube [18]

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

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

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

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

<table>
<thead>
<tr>
<th>Listing 10: pstricks.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>\def\Picture#1{%</td>
</tr>
<tr>
<td>\def\stripH[#1]{%</td>
</tr>
<tr>
<td>\begin{pspicture}[showgrid]</td>
</tr>
<tr>
<td>\psforeach{\row}{%</td>
</tr>
<tr>
<td>{{3,2.8,2.7,3,3.1}},%</td>
</tr>
<tr>
<td>{2.8,1.2,2,3},%</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>}%</td>
</tr>
<tr>
<td>\expandafter...</td>
</tr>
<tr>
<td>}</td>
</tr>
<tr>
<td>\end{pspicture}}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 11: pstricks.tex default output</th>
</tr>
</thead>
<tbody>
<tr>
<td>\def\Picture#1{%</td>
</tr>
<tr>
<td>\def\stripH[#1]{%</td>
</tr>
<tr>
<td>\begin{pspicture}[showgrid]</td>
</tr>
<tr>
<td>\psforeach{\row}{%</td>
</tr>
<tr>
<td>{{3,2.8,2.7,3,3.1}},%</td>
</tr>
<tr>
<td>{2.8,1.2,2,3},%</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>}%</td>
</tr>
<tr>
<td>\expandafter...</td>
</tr>
<tr>
<td>}</td>
</tr>
<tr>
<td>\end{pspicture}}</td>
</tr>
</tbody>
</table>
SECTION 3

How to use the script

latexindent.pl ships as part of the TEXLive distribution for Linux and Mac users; latexindent.exe ships as part of the TEXLive and MiKTeX distributions for Windows users. These files are also available from github [8] should you wish to use them without a TEX distribution; in this case, you may like to read appendix B on page 125 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 24.

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 123 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

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

This will output only the version number to the terminal.

- h, -help

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

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

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

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

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

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

This will indent `myfile.tex` and output it to `output.tex`, overwriting it 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).

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
$ latexindent.pl myfile.tex -o=output
$ 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
$ latexindent.pl myfile.tex -o=new
$ 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
$ 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
$ 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

```bash
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

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

See appendix D on page 128 for details of how the interface has changed from Version 2.2 to Version 3.0 for this flag.

-s, -silent

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

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

-t, -trace

```bash
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

```bash
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).

-1, -local[=myyaml.yaml,other.yaml,...]

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

latexindent.pl will always load defaultSettings.yaml (rhymes with camel) and if it is called with the -1 switch and it finds localSettings.yaml in the same directory as myfile.tex then these settings will be added to the indentation scheme. Information will be given in indent.log on the success or failure of loading localSettings.yaml.
The `-l` flag can take an optional parameter which details the name (or names separated by commas) of a YAML file(s) that resides in the same directory as `myfile.tex`; you can use this option if you would like to load a settings file in the current working directory that is not called `localSettings.yaml`. In fact, you can specify both relative and absolute paths for your YAML files; for example:

```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
cmh:~$ latexindent.pl -l="/uni2423/myyaml.yaml" myfile.tex
cmh:~$ latexindent.pl -l=myyaml.yaml+ myfile.tex
```

which translate, respectively, to

```bash
cmh:~$ latexindent.pl -l=localSettings.yaml,myyaml.yaml myfile.tex
cmh:~$ latexindent.pl -l=localSettings.yaml,myyaml.yaml myfile.tex
cmh:~$ latexindent.pl -l=myyaml.yaml,localSettings.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: 11:1"
cmh:~$ latexindent.pl myfile.tex -y="defaultIndent: 11:1",maximumIndentation: 3"
cmh:~$ latexindent.pl myfile.tex -y="indentRules: one: 11:1\t\t\t\t11:1"
cmh:~$ latexindent.pl myfile.tex
   -y="modifyLineBreaks:environments:EndStartsOnOwnLine:3" -m
cmh:~$ latexindent.pl myfile.tex
   -y="modifyLineBreaks:environments:one:EndStartsOnOwnLine:3" -m
```

You can specify YAML settings from the command line using the `-y` or `--yaml` switch; sample demonstrations are given above. Note, in particular, that multiple settings can be specified by separating
From the command line

them via commas. There is a further option to use a ; to separate fields, which is demonstrated in Section 4.3 on page 21.

Any settings specified via this switch will be loaded after any specified using the -l switch. This is discussed further in Section 4.4 on page 22.

-d, –onlydefault

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

-c, –cruft=<directory>

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.

-g, –logfile=<name of log file>

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

-sl, –screenlog

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

-m, –modifylinebreaks

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 66

latexindent.pl can also be called on a file without the file extension, for example
and in which case, you can specify the order in which extensions are searched for; see Listing 15 on page 24 for full details.

**STDIN**

```bash
$ cat myfile.tex | latexindent.pl
$ 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
$ 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
$ 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 [3] for an update to this feature.

### `-r`, `-replacement`

```bash
$ latexindent.pl -r myfile.tex
$ latexindent.pl -replacement 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 109.

### `-rv`, `-replacementrespectverb`

```bash
$ latexindent.pl -rv myfile.tex
$ latexindent.pl -replacementrespectverb 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 109.

### `-rr`, `-onlyreplacement`

```bash
$ latexindent.pl -rr myfile.tex
$ latexindent.pl -onlyreplacement myfile.tex
```

You can instruct latexindent.pl to skip all of its other indentation operations and *only* perform replacements/substitutions by using the `-rr` switch; full details and examples are given in Section 7 on page 109.
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 [2].
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 [6] 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

classpaths:
 - /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.

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

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

4.2 localSettings.yaml

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

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

then `latexindent.pl` will (assuming it exists) load `localSettings.yaml` from the same directory as `myfile.tex`.

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,

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

Any settings file(s) specified using the `-l` switch will be read after `defaultSettings.yaml` and, assuming they exist, any user setting files specified in `indentconfig.yaml`.

Your settings file can contain any switches that you’d like to change; a sample is shown in Listing 14, and you’ll find plenty of further examples throughout this manual.

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

4.3 The `-y|yaml` switch

You may use the `-y` switch to load your settings; for example, if you wished to specify the settings from Listing 14 using the `-y` switch, then you could use the following command:

```bash
cmh:$ latexindent.pl -y localSettings.yaml myfile.tex
```

Windows users may find that they have to end `.yaml` files with a blank line.

---

3Windows users may find that they have to end `.yaml` files with a blank line.
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.

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

```
cmh:~$ latexindent.pl -y="verbatimEnvironments:cmhenvironment:0;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,

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

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

If you wish to specify any regex-based settings using the -y switch, it is important not to use quotes surrounding the regex; for example, with reference to the 'one sentence per line' feature (Section 6.2 on page 76) and the listings within Listing 275 on page 78, the following settings give the option to have sentences end with a semicolon

```
cmh:~$ latexindent.pl -m --yaml=/quotesingle.ts1
   --yaml='modifyLineBreaks:oneSentencePerLine:sentencesEndWith:other:;'
```

A visual representation of this is given in Figure 1.
4.4 Settings load order

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.

### fileExtensionPreference: (fields)

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

```
cmh:~$ 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.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>41</td>
<td>fileExtensionPreference:</td>
</tr>
<tr>
<td>42</td>
<td>.tex: 1</td>
</tr>
<tr>
<td>43</td>
<td>.sty: 2</td>
</tr>
<tr>
<td>44</td>
<td>.cls: 3</td>
</tr>
<tr>
<td>45</td>
<td>.bib: 4</td>
</tr>
</tbody>
</table>

Calling latexindent.pl myfile with the (default) settings specified in Listing 15 means that the script will first look for myfile.tex, then myfile.sty, myfile.cls, and finally myfile.bib in order.

### backupExtension: (extension name)

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

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

### 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.

---

4Throughout this manual, listings shown with line numbers represent code taken directly from defaultSettings.yaml.
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.

\begin{verbatim}
  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
\end{verbatim}

The default value of cycleThroughBackUps is 0.

logFilePreferences: (fields)

\begin{verbatim}
85 logFilePreferences:
86   showEveryYamlRead: 1
87   showAmalgamatedSettings: 0
88   showDecorationStartCodeBlockTrace: 0
89   showDecorationFinishCodeBlockTrace: 0
90   endLogFileWith: '--------------'
91   showGitHubInfoFooter: 1
92   PatternLayout:
93     default: "%A%n"
94     trace: "%A%n"
95     ttrace: "%A%n"
\end{verbatim}

When either of the trace modes (see page 15) 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 127.

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.


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 17.

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 66).

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 19.

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 20 for example.

Trailing white space can be removed both before and after processing the document, as detailed in Listing 21; each of the fields can take the values 0 or 1. See Listings 386 to 388 on pages 98–99 for before and after results. Thanks to [19] for providing this feature.

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 22.
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 \texttt{fileContentsEnvironments}, see Listing 23. The behaviour of latexindent.pl on these environments is determined by their location (preamble or not), and the value \texttt{indentPreamble}, discussed next.

\begin{Verbatim}
\begin{lstlisting}
fileContentsEnvironments:
  filecontents: 1
  filecontents*: 1
\end{lstlisting}
\end{Verbatim}

\texttt{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 \texttt{indentPreamble} is set to 0, by default), but if you’d like latexindent.pl to try then change \texttt{indentPreamble} to 1.

\texttt{lookForPreamble: \{fields\}}

Not all files contain preamble; for example, sty, cls and bib files typically do not. Referencing Listing 24, if you set, for example, .tex to 0, then regardless of the setting of the value of \texttt{indentPreamble}, preamble will not be assumed when operating upon .tex files.

\begin{Verbatim}
\begin{lstlisting}
lookForPreamble:
  .tex: 1
  .sty: 0
  .cls: 0
  .bib: 0
\end{lstlisting}
\end{Verbatim}

\texttt{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 25.

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

\texttt{defaultIndent: \{horizontal space\}}

This is the default indentation (\tn means a tab, and is the default value) used in the absence of other details for the command or environment we are working with; see indentRules in Section 5.4 on page 43 for more details.

If you’re interested in experimenting with latexindent.pl then you can remove all indentation by setting \texttt{defaultIndent: ""}. 

lookForAlignDelims: \{fields\}

This contains a list of environments and/or commands that are operated upon in a special way by latexindent.pl (see Listing 26). In fact, the fields in lookForAlignDelims can actually take two different forms: the basic version is shown in Listing 26 and the advanced version in Listing 29; we will discuss each in turn.

**Listing 26: lookForAlignDelims (basic)**

<table>
<thead>
<tr>
<th>lookForAlignDelims:</th>
</tr>
</thead>
<tbody>
<tr>
<td>tabular: 1</td>
</tr>
<tr>
<td>tabularx: 1</td>
</tr>
<tr>
<td>longtable: 1</td>
</tr>
<tr>
<td>array: 1</td>
</tr>
<tr>
<td>matrix: 1</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

The environments specified in this field will be operated on in a special way by latexindent.pl. In particular, it will try and align each column by its alignment tabs. It does have some limitations (discussed further in Section 9), but in many cases it will produce results such as those in Listings 27 and 28.

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 19 on page 26).

**Listing 27: tabular1.tex**

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

**Listing 28: tabular1.tex default output**

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

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 29 is for you.

**Listing 29: lookForAlignDelims (advanced)**

```
lookForAlignDelims: 
  tabular: 1 
  delims: 1 
  alignDoubleBackSlash: 1 
  spacesBeforeDoubleBackSlash: 1 
  multiColumnGrouping: 0 
  alignRowsWithoutMaxDelims: 1 
  spacesBeforeAmpersand: 1 
  spacesAfterAmpersand: 1 
  justification: left 
  alignFinalDoubleBackSlash: 0 
  dontMeasure: 0 
  delimiterRegEx: '\(\langle!\\\\&\)\' 
  delimiterJustification: left 
  tabularx: 
    delims: 1 
  longtable: 1 
```

Note that you can use a mixture of the basic and advanced form: in Listing 29 \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 26. 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). \(^5\)

• `multiColumnGrouping`: binary switch (0 or 1) that details if \texttt{latexindent.pl} should group columns above and below a \texttt{\multicolumn} command (default: 0);

• `alignRowsWithoutMaxDelims`: binary switch (0 or 1) that details if rows that do not contain the maximum number of delimiters should be formatted so as to have the ampersands aligned (default: 1);

• `spacesBeforeAmpersand`: optionally specifies the number (integer ≥ 0) of spaces to be placed before ampersands (default: 1);

• `spacesAfterAmpersand`: optionally specifies the number (integer ≥ 0) of spaces to be placed after ampersands (default: 1);

• `justification`: optionally specifies the justification of each cell as either \textit{left} or \textit{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 delimeter (default: `'(?<!\)(&)'`);

• `delimiterJustification` optionally specifies the justification for the alignment delimeters (default: left); note that this feature is only useful if you have delimiters of different lengths in the same column, discussed in Section 5.2.

We will explore most of these features using the file \texttt{tabular2.tex} in Listing 30 (which contains a \texttt{\multicolumn} command), and the YAML files in Listings 31 to 37; we will explore `alignFinalDoubleBackSlash` in Listing 46; the `dontMeasure` feature will be described in Section 5.1, and `delimiterRegEx` in Section 5.2.

\begin{verbatim}
\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}
\end{verbatim}

\begin{verbatim}
lookForAlignDelims: tabular:
  multiColumnGrouping: 1
\end{verbatim}

\begin{verbatim}
lookForAlignDelims: tabular:
  alignRowsWithoutMaxDelims: 0
\end{verbatim}

\(^5\)Previously this only activated if `alignDoubleBackSlash` was set to 0.
On running the commands

```
cmh:~$ latexindent.pl tabular2.tex

cmh:~$ latexindent.pl tabular2.tex -l tabular2.yaml

cmh:~$ latexindent.pl tabular2.tex -l tabular3.yaml

cmh:~$ latexindent.pl tabular2.tex -l tabular2.yaml,tabular4.yaml

```

we obtain the respective outputs given in Listings 38 to 45.

**LISTING 38: tabular2.tex default output**

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

**LISTING 39: tabular2.tex using Listing 31**

\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 40: tabular2.tex using Listing 32

\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 41: tabular2.tex using Listings 31 and 33

\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 42: tabular2.tex using Listings 31 and 34

\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 43: tabular2.tex using Listings 31 and 35

\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 44: tabular2.tex using Listings 31 and 36

\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}
\begin{tabular}{cccc}
  A & B & C & D \\
  AAA & BBB & CCC & DDD \\
\end{tabular}

\begin{tabular}{cc}
\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 38 and 39 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 38 the columns have been aligned at the ampersand;
• in Listing 39 the \texttt{\textbackslash multicolumn} command has grouped the 2 columns beneath and above it, because \texttt{multiColumnGrouping} is set to 1 in Listing 31;
• in Listing 40 rows 3 and 6 have not been aligned at the ampersand, because \texttt{alignRowsWithoutMaxDelims} has been set to 0 in Listing 32; however, the \texttt{\textbackslash \textbackslash} have still been aligned;
• in Listing 41 the columns beneath and above the \texttt{\textbackslash multicolumn} commands have been grouped (because \texttt{multiColumnGrouping} is set to 1), and there are at least 4 spaces before each aligned ampersand because \texttt{spacesBeforeAmpersand} is set to 4;
• in Listing 42 the columns beneath and above the \texttt{\textbackslash multicolumn} commands have been grouped (because \texttt{multiColumnGrouping} is set to 1), and there are at least 4 spaces after each aligned ampersand because \texttt{spacesAfterAmpersand} is set to 4;
• in Listing 43 the \texttt{\textbackslash \textbackslash} have not been aligned, because \texttt{alignDoubleBackSlash} is set to 0, otherwise the output is the same as Listing 39;
• in Listing 44 the \texttt{\textbackslash \textbackslash} have been aligned, and because \texttt{spacesBeforeDoubleBackSlash} is set to 0, there are no spaces ahead of them; the output is otherwise the same as Listing 39.
• in Listing 45 the cells have been right-justified; note that cells above and below the \texttt{\textbackslash multicolumn} statements have still been grouped correctly, because of the settings in Listing 31.

We explore the \texttt{alignFinalDoubleBackSlash} feature by using the file in Listing 46. Upon running the following commands

\begin{verbatim}
  cmh:~$ latexindent.pl tabular4.tex -o=+-default
  cmh:~$ latexindent.pl tabular4.tex -o=+-FDBS
  -y="lookForAlignDelims:tabular:alignFinalDoubleBackSlash:1"
\end{verbatim}

then we receive the respective outputs given in Listing 47 and Listing 48.

\begin{tabular}{lc}
Name & \texttt{\textbackslash shortstack}\texttt{Hi} \texttt{\textbackslash \textbackslash} \texttt{Lo} \texttt{\textbackslash \textbackslash} \\
Foo & Bar \\
\end{tabular}
As of Version 3.0, the alignment routine works on mandatory and optional arguments within commands, and also within 'special' code blocks (see specialBeginEnd on page 37); for example, assuming that you have a command called \matrix and that it is populated within lookForAlignDelims (which it is, by default), and that you run the command

```
cmh:~> latexindent.pl matrix1.tex
```

then the before-and-after results shown in Listings 49 and 50 are achievable by default.

<table>
<thead>
<tr>
<th>Listing 49: matrix1.tex</th>
<th>Listing 50: matrix1.tex default output</th>
</tr>
</thead>
<tbody>
<tr>
<td>\matrix [</td>
<td>\matrix [</td>
</tr>
<tr>
<td>1&amp;2 &amp; 3 &amp; &amp; &amp; 4 &amp;</td>
<td>1 &amp; 2 &amp; 3 &amp;</td>
</tr>
<tr>
<td>7&amp;8 &amp; 9 &amp;</td>
<td>4 &amp; 5 &amp; 6 &amp;</td>
</tr>
<tr>
<td>10&amp;11&amp;12</td>
<td>7 &amp; 8 &amp; 9 &amp;</td>
</tr>
<tr>
<td>]</td>
<td>10 &amp; 11 &amp; 12</td>
</tr>
</tbody>
</table>

If you have blocks of code that you wish to align at the & character that are not wrapped in, for example, \begin{tabular}...\end{tabular}, then you can use the mark up illustrated in Listing 51; the default output is shown in Listing 52. Note that the %* must be next to each other, but that there can be any number of spaces (possibly none) between the * and \begin{tabular}; note also that you may use any environment name that you have specified in lookForAlignDelims.

<table>
<thead>
<tr>
<th>Listing 51: align-block.tex</th>
<th>Listing 52: align-block.tex default output</th>
</tr>
</thead>
<tbody>
<tr>
<td>%* \begin{tabular}</td>
<td>%* \begin{tabular}</td>
</tr>
<tr>
<td>1 &amp; 2 &amp; 3 &amp; 4 &amp;</td>
<td>1 &amp; 2 &amp; 3 &amp; 4 &amp;</td>
</tr>
<tr>
<td>5 &amp; 6 &amp;</td>
<td>4 &amp; 5 &amp; 6 &amp;</td>
</tr>
<tr>
<td>%* \end{tabular}</td>
<td>%* \end{tabular}</td>
</tr>
</tbody>
</table>

With reference to Table 1 on page 44 and the, yet undiscussed, fields of noAdditionalIndent and indentRules (see Section 5.4 on page 43), these comment-marked blocks are considered environments.

5.1 lookForAlignDelims: the dontMeasure feature

The lookForAlignDelims field can, optionally, receive the dontMeasure option which can be specified in a few different ways. We will explore this feature in relation to the code given in Listing 53; the default output is shown in Listing 54.

<table>
<thead>
<tr>
<th>Listing 53: tabular-DM.tex</th>
<th>Listing 54: tabular-DM.tex default output</th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{tabular}{cccc}</td>
<td>\begin{tabular}{cccc}</td>
</tr>
<tr>
<td>aaaaa &amp; bbbbb &amp; ccc &amp; dd |</td>
<td>aaaaa &amp; bbbbb &amp; ccc &amp; dd |</td>
</tr>
<tr>
<td>11 &amp; 2 &amp; 33 &amp; 4 |</td>
<td>11 &amp; 2 &amp; 33 &amp; 4 |</td>
</tr>
<tr>
<td>5 &amp; 66 &amp; 7&amp;8 |</td>
<td>5 &amp; 66 &amp; 7 &amp; 8 |</td>
</tr>
<tr>
<td>\end{tabular}</td>
<td>\end{tabular}</td>
</tr>
</tbody>
</table>

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 56, we can run the command

```
cmh:~> latexindent.pl tabular-DM.tex -l=dontMeasure1.yaml
```
and receive the output given in Listing 55.
5.1 lookForAlignDelims: the dontMeasure feature

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 58. On running the following commands,

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

we receive the output in Listing 57.

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

We note that in Listing 58 we have specified entries not to be measured, one entry per line.

The dontMeasure field can also be specified in the forms demonstrated in Listing 60 and Listing 61. Upon running the commands

```
cmh:~$ latexindent.pl tabular-DM.tex -l= dontMeasure3.yaml
cmh:~$ latexindent.pl tabular-DM.tex -l= dontMeasure4.yaml
```

we receive the output given in Listing 59.

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

We note that in:

- Listing 60 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 61 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 63; upon
5.2 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: a 

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 66; the default output from latexindent.pl is given in Listing 67.

Let’s say that we wish to align the code at either the \ or \>. We employ the settings given in Listing 69 and run the command

cmh:\$ latexindent.pl tabbing.tex -l=delimiterRegEx1.yaml

to receive the output given in Listing 68.
5.2 lookForAlignDelims: the delimiterRegEx and delimiterJustification feature

We note that:

• in Listing 68 the code has been aligned, as intended, at both the \= and \>;

• in Listing 69 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 71 and run the command

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

we receive the output in Listing 75.

```
\begin{tabbing}
\alpha=bb=\gamma=dd=ee \ \\
\>|2>|1>|7>|3| \\
\>|3>|2>|8>|3| \\
\>|4>|2| \end{tabbing}
```
5.2 lookForAlignDelims: the delimiterRegEx and delimiterJustification feature

You can set the delimiter justification as either left (default) or right, which will only have effect when delimiters in the same column have different lengths. Using the settings in Listing 78 and running the command

cmh:~$ latexindent.pl tabbing1.tex -l=delimiterRegEx5.yaml -o=+-mod5

gives the output in Listing 77.

Note that in Listing 77 the second set of delimiters have been right aligned – it is quite subtle!

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 80 and 81.

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 20 for details of how to configure user settings, and Listing 13 on page 21 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 83 shows the default settings of `specialBeginEnd`.

```
<table>
<thead>
<tr>
<th>Listing 83: specialBeginEnd</th>
</tr>
</thead>
<tbody>
<tr>
<td>207  specialBeginEnd:</td>
</tr>
<tr>
<td>208    displayMath:</td>
</tr>
<tr>
<td>209      begin: '\['</td>
</tr>
<tr>
<td>210      end: '\]'</td>
</tr>
<tr>
<td>211    lookForThis: 1</td>
</tr>
<tr>
<td>212  inlineMath:</td>
</tr>
<tr>
<td>213      begin: '((?&lt;!$)(?&lt;!$)$(?!$)'</td>
</tr>
<tr>
<td>214      end: '((?&lt;!$)$(?!$)'</td>
</tr>
<tr>
<td>215    lookForThis: 1</td>
</tr>
<tr>
<td>216  displayMathTex:</td>
</tr>
<tr>
<td>217      begin: '\$'</td>
</tr>
<tr>
<td>218      end: '\$'</td>
</tr>
<tr>
<td>219    lookForThis: 1</td>
</tr>
<tr>
<td>220  specialBeforeCommand: 0</td>
</tr>
</tbody>
</table>
```

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 21); 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 84 and 85.

```
<table>
<thead>
<tr>
<th>Listing 84: special1.tex before</th>
<th>Listing 85: special1.tex default output</th>
</tr>
</thead>
<tbody>
<tr>
<td>The function $f$ has formula</td>
<td>The function $f$ has formula</td>
</tr>
<tr>
<td>[ f(x)=x^2. ]</td>
<td>[ f(x)=x^2. ]</td>
</tr>
<tr>
<td>If you like splitting dollars,</td>
<td>If you like splitting dollars,</td>
</tr>
<tr>
<td>$g(x)=f(2x)$</td>
<td>$g(x)=f(2x)$</td>
</tr>
</tbody>
</table>
```

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 86.

```
<table>
<thead>
<tr>
<th>Listing 86: specialLR.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{equation}</td>
</tr>
<tr>
<td>\left[ \sqrt{a+b} \right]</td>
</tr>
<tr>
<td>\end{equation}</td>
</tr>
</tbody>
</table>
```

Now consider the YAML files shown in Listings 87 and 88.

```
<table>
<thead>
<tr>
<th>Listing 87: specialsLeftRight.yaml</th>
<th>Listing 88: specialBeforeCommand.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>specialBeginEnd:</td>
<td>specialBeginEnd:</td>
</tr>
<tr>
<td>leftRightSquare:</td>
<td>specialBeforeCommand: 1</td>
</tr>
<tr>
<td>begin: '\left[\right]'</td>
<td></td>
</tr>
<tr>
<td>end: '\left[\right]'</td>
<td></td>
</tr>
<tr>
<td>lookForThis: 1</td>
<td></td>
</tr>
</tbody>
</table>
```
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 89 and 90.

<table>
<thead>
<tr>
<th>LISTING 89: specialLR.tex using Listing 87</th>
<th>LISTING 90: specialLR.tex using Listings 87 and 88</th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{equation} \left[ \sqrt{a+b} \right] \end{equation}</td>
<td>\begin{equation} \left[ \sqrt{a+b} \right] \end{equation}</td>
</tr>
</tbody>
</table>

Notice that in:

- Listing 89 the `\left` has been treated as a *command*, with one optional argument;
- Listing 90 the `specialBeginEnd` pattern in Listing 87 has been obeyed because Listing 88 specifies that the `specialBeginEnd` should be sought *before* commands.

You can, optionally, specify the *middle* field for anything that you specify in `specialBeginEnd`. For example, let's consider the `.tex` file in Listing 91.

```
\begin{If} something 0 \\ElseIf something 1 \\ElseIf something 2 \\ElseIf something 3 \Else something 4 \\EndIf
```

Upon saving the YAML settings in Listings 92 and 94 and running the commands

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

then we obtain the output given in Listings 93 and 95.
We note that:

- in Listing 93 the bodies of each of the Elsif statements have been indented appropriately;
- the Else statement has not been indented appropriately in Listing 93 – read on!
- we have specified multiple settings for the middle field using the syntax demonstrated in Listing 94 so that the body of the Else statement has been indented appropriately in Listing 95.

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 97 and the YAML in Listing 96, and running

```
cmh:~$ latexindent.pl special3.tex -l=special-verb1
```

then the output in Listing 97 is unchanged.

```
Listing 97: special3.tex and output using Listing 96
\[
\text{special code blocks can be treated as verbatim}\]
```

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
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 5.4 on page 43); 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 99 saved into `headings1.yaml`, and that you have the text from Listing 100 saved into `headings1.tex`.

If you run the command

```
$ latexindent.pl headings1.tex -l=headings1.yaml
```

then you should receive the output given in Listing 101.

---

6There is a slight difference in interface for this field when comparing Version 2.2 to Version 3.0; see appendix D on page 128 for details.
Now say that you modify the YAML from Listing 99 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 102; 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 [16], and is off by default.

For example, consider the example shown in Listing 103 together with the default output shown in Listing 104.

<table>
<thead>
<tr>
<th>Listing 103: mult-nested.tex</th>
<th>Listing 104: mult-nested.tex default output</th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{one} one \begin{two} two \begin{three} three \begin{four} four \end{four} \end{three} \end{two} \end{one}</td>
<td>\begin{one} %one \begin{two} %two \begin{three} %three \begin{four} %four \end{four} \end{three} \end{two} \end{one}</td>
</tr>
</tbody>
</table>

Now say that, for example, you have the max-indentation1.yaml from Listing 105 and that you run the following command:

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

You should receive the output shown in Listing 106.

<table>
<thead>
<tr>
<th>Listing 105: max-indentation1.yaml</th>
<th>Listing 106: mult-nested.tex using Listing 105</th>
</tr>
</thead>
<tbody>
<tr>
<td>maximumIndentation: &quot; &quot;</td>
<td>\begin{one} %one \begin{two} %two \begin{three} %three \begin{four} %four \end{four} \end{three} \end{two} \end{one}</td>
</tr>
</tbody>
</table>

Comparing the output in Listings 104 and 106 we notice that the (default) tabs of indentation have been replaced by a single space.

In general, when using the maximumIndentation feature, any leading tabs will be replaced by equivalent spaces except, of course, those found in verbatimEnvironments (see Listing 17 on page 26)
or noIndentBlock (see Listing 19 on page 26).

5.3 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 118.

5.4 noAdditionalIndent and indentRules

latexindent.pl operates on files by looking for code blocks, as detailed in Section 5.3; for each type of code block in Table 1 on the following page (which we will call a \textit{thing} in what follows) it searches YAML fields for information in the following order:

1. noAdditionalIndent for the \textit{name} of the current \textit{thing};
2. indentRules for the \textit{name} of the current \textit{thing};
3. noAdditionalIndentGlobal for the \textit{type} of the current \textit{thing};
4. indentRulesGlobal for the \textit{type} of the current \textit{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 next page; for reference, there follows a list of the code blocks covered.

5.4.1 Environments and their arguments ........................................... 43
5.4.2 Environments with items ....................................................... 51
5.4.3 Commands with arguments ..................................................... 52
5.4.4 ifelsefi code blocks ............................................................. 54
5.4.5 specialBeginEnd code blocks .................................................. 55
5.4.6 afterHeading code blocks ...................................................... 56
5.4.7 The remaining code blocks ..................................................... 58
   keyEqualsValuesBracesBrackets ............................................... 58
   namedGroupingBracesBrackets .................................................. 59
   UnNamedGroupingBracesBrackets .............................................. 59
   filecontents ................................................................. 60
5.4.8 Summary ................................................................. 60

5.4.1 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 107.

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

\begin{verbatim}
\begin{outer}
  \begin{myenv}
    \text{body of environment}
    \text{body of environment}
    \text{body of environment}
  \end{myenv}
\end{outer}
\end{verbatim}
### Table 1: Code blocks known to \texttt{latexindent.pl}

<table>
<thead>
<tr>
<th>Code block</th>
<th>characters allowed in name</th>
<th>example</th>
</tr>
</thead>
<tbody>
<tr>
<td>environments</td>
<td>a-zA-Z0*0-9_\</td>
<td>\begin{myenv} body of myenv \end{myenv}</td>
</tr>
<tr>
<td>optionalArguments</td>
<td>\textit{inherits} name from parent (e.g. environment name)</td>
<td>\begin{myenv} body of myenv \end{myenv}</td>
</tr>
<tr>
<td>mandatoryArguments</td>
<td>\textit{inherits} name from parent (e.g. environment name)</td>
<td>\begin{myenv} body of myenv \end{myenv}</td>
</tr>
<tr>
<td>commands</td>
<td>+a-zA-Z0*0-9_\</td>
<td>\begin{myenv} arguments \end{myenv}</td>
</tr>
<tr>
<td>keyEqualsValuesBracesBrackets</td>
<td>a-zA-Z0*0-9_\</td>
<td>\begin{myenv} arguments \end{myenv}</td>
</tr>
<tr>
<td>namedGroupingBracesBrackets</td>
<td>0-9_.a-zA-Z0**&lt;</td>
<td>\begin{myenv} arguments \end{myenv}</td>
</tr>
<tr>
<td>UnNamedGroupingBracesBrackets</td>
<td>\textit{No name!}</td>
<td>\begin{myenv} arguments \end{myenv}</td>
</tr>
<tr>
<td>ifElseFi</td>
<td>@a-zA-Z but must begin with either</td>
<td>\begin{myenv} arguments \end{myenv}</td>
</tr>
<tr>
<td>items</td>
<td>User specified, see Listings 79 and 82 on page 37</td>
<td>\begin{myenv} arguments \end{myenv}</td>
</tr>
<tr>
<td>specialBeginEnd</td>
<td>User specified, see Listing 83 on page 38</td>
<td>\begin{myenv} arguments \end{myenv}</td>
</tr>
<tr>
<td>afterHeading</td>
<td>User specified, see Listing 98 on page 41</td>
<td>\begin{myenv} arguments \end{myenv}</td>
</tr>
<tr>
<td>filecontents</td>
<td>User specified, see Listing 23 on page 27</td>
<td>\begin{myenv} arguments \end{myenv}</td>
</tr>
</tbody>
</table>
If we do not wish `myenv` to receive any additional indentation, we have a few choices available to us, as demonstrated in Listings 108 and 109.

### Listing 108: `myenv-noAdd1.yaml`

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

### Listing 109: `myenv-noAdd2.yaml`

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

On applying either of the following commands,

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

we obtain the output given in Listing 110; note in particular that the environment `myenv` has not received any additional indentation, but that the outer environment has still received indentation.

### Listing 110: `myenv.tex` output (using either Listing 108 or Listing 109)

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

Upon changing the YAML files to those shown in Listings 111 and 112, 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 113.

### Listing 111: `myenv-noAdd3.yaml`

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

### Listing 112: `myenv-noAdd4.yaml`

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

### Listing 113: `myenv.tex` output (using either Listing 111 or Listing 112)

```latex
\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 114.
Upon running

```bash
$ latexindent -l myenv-noAdd1.yaml myenv-args.tex
```

we obtain the output shown in Listing 115; 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 108), 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 116 and 117.

Upon running

```bash
$ latexindent -l myenv-noAdd5.yaml myenv-args.tex
$ latexindent -l myenv-noAdd6.yaml myenv-args.tex
```

we obtain the respective outputs given in Listings 118 and 119. Note that in Listing 118 the text for the optional argument has not received any additional indentation, and that in Listing 119 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 120 and 121.

On applying either of the following commands,

```
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 122; note in particular that the environment `myenv` has received one tab (from the outer environment) plus three spaces from Listing 120 or 121.

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

Returning to the example in Listing 114 that contains optional and mandatory arguments. Upon using Listing 120 as in

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

we obtain the output in Listing 123; 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 124 and 125.

**LISTING 124: myenv-rules3.yaml**

```yaml
indentRules:
  myenv:
    body: " "
    optionalArguments: " "
```

**LISTING 125: myenv-rules4.yaml**

```yaml
indentRules:
  myenv:
    body: " "
    mandatoryArguments: "\t\t"
```

After running

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

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

then we obtain the respective outputs given in Listings 126 and 127.

**LISTING 126: myenv-args.tex using Listing 124**

```latex
\begin{outer}
  \begin{myenv}[%
    \optionalArgument{text}
    \optionalArgument{text}]
    \mandatoryArgument{text}
  \end{myenv}
\end{outer}
```

**LISTING 127: myenv-args.tex using Listing 125**

```latex
\begin{outer}
  \begin{myenv}[%
    \optionalArgument{text}
    \optionalArgument{text}]
    \mandatoryArgument{text}
  \end{myenv}
\end{outer}
```

Note that in Listing 126, 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 127, 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 List-
Let's say that you change the value of environments to 1 in Listing 128, and that you run

```
$ latexindent.pl myenv-args.tex -l env-noAdditionalGlobal.yaml
$ latexindent.pl myenv-args.tex -l myenv-rules1.yaml,env-noAdditionalGlobal.yaml
```

The respective output from these two commands are in Listings 129 and 130; in Listing 129 notice that both environments receive no additional indentation but that the arguments of `myenv` still do receive indentation. In Listing 130 notice that the `outer` environment does not receive additional indentation, but because of the settings from `myenv-rules1.yaml` (in Listing 120 on page 47), the `myenv` environment still does receive indentation.

```
\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}
```

In fact, `noAdditionalIndentGlobal` also contains keys that control the indentation of optional and mandatory arguments; on referencing Listings 131 and 132

```
\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}
```

we may run the commands

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

which produces the respective outputs given in Listings 133 and 134. Notice that in Listing 133 the optional argument has not received any additional indentation, and in Listing 134 the mandatory argument has not received any additional indentation.

```
\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 135; 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 136 and 137. Note that in Listing 136, both the environment blocks have received a single-space indentation, whereas in Listing 137 the outer environment has received single-space indentation (specified by `indentRulesGlobal`), but `myenv` has received " ", as specified by the particular `indentRules` for `myenv` Listing 120 on page 47.

```
\begin{outer}
  \begin{myenv}[%
    \text{optional_argument_text}
    \text{mandatory_argument_text}]
  \text{body_of_environment}
  \end{myenv}
  \end{outer}
```

You can specify `indentRulesGlobal` for both optional and mandatory arguments, as detailed in Listings 138 and 139:

```
\begin{outer}
  \begin{myenv}[%
    \text{optional_argument_text}
    \text{mandatory_argument_text}]
  \text{body_of_environment}
  \end{myenv}
  \end{outer}
```

Upon running the following commands:

```bash
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 140 and 141. Note that the `optional` argument in Listing 140 has received two tabs worth of indentation, while the `mandatory` argument has done so in Listing 141.
5.4.2 Environments with items

With reference to Listings 79 and 82 on page 37, some commands may contain \texttt{item} commands; for the purposes of this discussion, we will use the code from Listing 80 on page 37.

Assuming that you’ve populated \texttt{itemNames} with the name of your \texttt{item}, you can put the item name into \texttt{noAdditionalIndent} as in Listing 142, although a more efficient approach may be to change the relevant field in \texttt{itemNames} to 0. Similarly, you can customise the indentation that your \texttt{item} receives using \texttt{indentRules}, as in Listing 143.

Upon running the following commands:

\begin{Verbatim}
\texttt{cmh:}\$ \texttt{latexindent.pl \textit{items1.tex} -local item-noAdd1.yaml}
\texttt{cmh:}\$ \texttt{latexindent.pl \textit{items1.tex} -local item-rules1.yaml}
\end{Verbatim}

the respective outputs are given in Listings 144 and 145; note that in Listing 144 that the text after each \texttt{item} has not received any additional indentation, and in Listing 145, the text after each \texttt{item} has received a single space of indentation, specified by Listing 143.

Alternatively, you might like to populate \texttt{noAdditionalIndentGlobal} or \texttt{indentRulesGlobal} using the \texttt{items} key, as demonstrated in Listings 146 and 147. Note that there is a need to ‘reset/remove’ the \texttt{item} field from \texttt{indentRules} in both cases (see the hierarchy description given on page 43) as the \texttt{item} command is a member of \texttt{indentRules} by default.
Upon running the following commands,

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

the respective outputs from Listings 144 and 145 are obtained; note, however, that all such item commands without their own individual noAdditionalIndent or indentRules settings would behave as in these listings.

### 5.4.3 Commands with arguments

Let's begin with the simple example in Listing 148; when latexindent.pl operates on this file, the default output is shown in Listing 149.

**Listing 148: mycommand.tex**

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

**Listing 149: mycommand.tex default output**

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

As in the environment-based case (see Listings 108 and 109 on page 45) we may specify noAdditionalIndent either in ‘scalar’ form, or in ‘field’ form, as shown in Listings 150 and 151.

**Listing 150: mycommand-noAdd1.yaml**

```yaml
noAdditionalIndent:
  mycommand: 1
```

**Listing 151: mycommand-noAdd2.yaml**

```yaml
noAdditionalIndent:
  mycommand:
    body: 1
```

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 152 and 153.

**Listing 152: mycommand.tex using Listing 150**

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

**Listing 153: mycommand.tex using Listing 151**

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

Note that in Listing 152 that the ‘body’, optional argument and mandatory argument have all received no additional indentation, while in Listing 153, 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.

---

7The command code blocks have quite a few subtleties, described in Section 5.5 on page 60.
We may further customise `noAdditionalIndent` for `mycommand` as we did in Listings 116 and 117 on page 46; explicit examples are given in Listings 154 and 155.

**Listing 154:**
```
noAdditionalIndent:
    mycommand:
        body: 0
        optionalArguments: 1
        mandatoryArguments: 0
```

**Listing 155:**
```
noAdditionalIndent:
    mycommand:
        body: 0
        optionalArguments: 0
        mandatoryArguments: 1
```

After running the following commands,
```
cmh:~$ latexindent.pl mycommand.tex -l mycommand-noAdd3.yaml
```
```
cmh:~$ latexindent.pl mycommand.tex -l mycommand-noAdd4.yaml
```
we receive the respective output given in Listings 156 and 157.

**Listing 156:**
```
\mycommand
{ 
    mand arg text
    mand arg text}
[  
    opt arg text
    opt arg text
]
```

**Listing 157:**
```
\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 156 and 157 has received no additional indent, even though body is explicitly set to 0 in both Listings 154 and 155. This is because, by default, `noAdditionalIndentGlobal` for commands is set to 1 by default; this can be easily fixed as in Listings 158 and 159.

**Listing 158:**
```
noAdditionalIndent:
    mycommand:
        body: 0
        optionalArguments: 1
        mandatoryArguments: 0
        noAdditionalIndentGlobal:
            commands: 0
```

**Listing 159:**
```
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 160 and 161.
5.4 noAdditionalIndent and indentRules

Both indentRules and indentRulesGlobal can be adjusted as they were for environment code blocks, as in Listings 124 and 125 on page 48 and Listings 135, 138 and 139 on page 50.

5.4.4 ifelsefi code blocks

Let's use the simple example shown in Listing 162; when latexindent.pl operates on this file, the output as in Listing 163; 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 noAdditionalIndent and indentRules in the ‘scalar’ form only for these type of code blocks, although the ‘field’ form would work, assuming that body was specified. Examples are shown in Listings 164 and 165.

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 166 and 167; note that in Listing 166, the ifnum code block has not received any additional indentation, while in Listing 167, the ifnum code block has received one tab and two spaces of indentation.

We may specify noAdditionalIndentGlobal and indentRulesGlobal as in Listings 168 and 169.
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 170 and 171; notice that in Listing 170 neither of the `ifelsefi` code blocks have received indentation, while in Listing 171 both code blocks have received a single space of indentation.

We can further explore the treatment of `ifElseFi` code blocks in Listing 172, and the associated default output given in Listing 173; note, in particular, that the bodies of each of the ‘or statements’ have been indented.

After running the following commands,
we receive the respective output given in Listings 176 and 177; note that in Listing 176, the `displayMath` code block has not received any additional indentation, while in Listing 177, the `displayMath` code block has received three tabs worth of indentation.

**LISTING 176:** special1.tex using Listing 174

The function $f$ has formula
\[
\text{If you like splitting dollars, }$
\]
\[
g(x)=f(2x)
\]

**LISTING 177:** special1.tex using Listing 175

The function $f$ has formula
\[
\text{If you like splitting dollars, }$
\]
\[
g(x)=f(2x)
\]

We may specify `noAdditionalIndentGlobal` and `indentRulesGlobal` as in Listings 178 and 179.

**LISTING 178:** special-noAdd-glob.yaml

`noAdditionalIndentGlobal:`
`specialBeginEnd: 1`

**LISTING 179:** special-indent-rules-global.yaml

`indentRulesGlobal:`
`specialBeginEnd: " "`

Upon running the following commands

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

we receive the outputs in Listings 180 and 181; notice that in Listing 180 neither of the special code blocks have received indentation, while in Listing 181 both code blocks have received a single space of indentation.

**LISTING 180:** special1.tex using Listing 178

The function $f$ has formula
\[
\text{If you like splitting dollars, }$
\]
\[
g(x)=f(2x)
\]

**LISTING 181:** special1.tex using Listing 179

The function $f$ has formula
\[
\text{If you like splitting dollars, }$
\]
\[
g(x)=f(2x)
\]

### 5.4.6 afterHeading code blocks

Let's use the example Listing 182 for demonstration throughout this Section. As discussed on page 41, by default `latexindent.pl` will not add indentation after headings.

**LISTING 182:** headings2.tex

\begin{paragraph}{}
\text{paragraph text}
\text{paragraph text}
\end{paragraph}
we obtain the output in Listing 183. 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 186 and run the command

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

then we receive the output in Listing 185. 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 188 and run analogous commands to those above, we receive the output in Listing 187; 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 190 which gives the output in Listing 189.

Analogously, we may specify indentRules as in Listing 192 which gives the output in Listing 191; 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 194 and 196 respectively, with respective output in Listings 193 and 195. Note that in Listing 194 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 195, the argument has received both a (default) tab plus two spaces of indentation (from the global rule specified in Listing 196), and the remaining body after paragraph has received just two spaces of indentation.

5.4.7 The remaining code blocks

Referencing the different types of code blocks in Table 1 on page 44, we have a few code blocks yet to cover; these are very similar to the commands code block type covered comprehensively in Section 5.4.3 on page 52, but a small discussion defining these remaining code blocks is necessary.

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 44;
- 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 470 on page 118

An example is shown in Listing 197, with the default output given in Listing 198.
In Listing 198, 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 43.

**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 44;
- 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 470 on page 118

A simple example is given in Listing 199, with default output in Listing 200.

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

In particular, latexindent.pl considers child, parent and node all to be namedGroupingBracesBrackets. Referencing Listing 200, 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 43.

**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 ( 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 470 on page 118

An example is shown in Listing 201 with default output give in Listing 202.

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

---

8You may like to verify this by using the -tt option and checking indent.log!
Referencing Listing 202, 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 43.

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

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

### 5.4.8 Summary

Having considered all of the different types of code blocks, the functions of the fields given in Listings 203 and 204 should now make sense.

<table>
<thead>
<tr>
<th>Listing 203: noAdditionalIndentGlobal</th>
<th>Listing 204: indentRulesGlobal</th>
</tr>
</thead>
<tbody>
<tr>
<td>288 noAdditionalIndentGlobal:</td>
<td>304 indentRulesGlobal:</td>
</tr>
<tr>
<td>289 environments: 0</td>
<td>305 environments: 0</td>
</tr>
<tr>
<td>290 commands: 1</td>
<td>306 commands: 0</td>
</tr>
<tr>
<td>291 optionalArguments: 0</td>
<td>307 optionalArguments: 0</td>
</tr>
<tr>
<td>292 mandatoryArguments: 0</td>
<td>308 mandatoryArguments: 0</td>
</tr>
<tr>
<td>293 ifElseFi: 0</td>
<td>309 ifElseFi: 0</td>
</tr>
<tr>
<td>294 items: 0</td>
<td>310 items: 0</td>
</tr>
<tr>
<td>295 keyEqualsValuesBracesBrackets: 0</td>
<td>311 keyEqualsValuesBracesBrackets: 0</td>
</tr>
<tr>
<td>296 namedGroupingBracesBrackets: 0</td>
<td>312 namedGroupingBracesBrackets: 0</td>
</tr>
<tr>
<td>297 UnNamedGroupingBracesBrackets: 0</td>
<td>313 UnNamedGroupingBracesBrackets: 0</td>
</tr>
<tr>
<td>298 specialBeginEnd: 0</td>
<td>314 specialBeginEnd: 0</td>
</tr>
<tr>
<td>299 afterHeading: 0</td>
<td>315 afterHeading: 0</td>
</tr>
<tr>
<td>300 filecontents: 0</td>
<td>316 filecontents: 0</td>
</tr>
</tbody>
</table>

### 5.5 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.

The `commandCodeBlocks` field contains a few switches detailed in Listing 205.
5.5 Commands and the strings between their arguments

<table>
<thead>
<tr>
<th>Listing 205: commandCodeBlocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>commandCodeBlocks:</td>
</tr>
<tr>
<td>roundParenthesesAllowed: 1</td>
</tr>
<tr>
<td>stringsAllowedBetweenArguments:</td>
</tr>
<tr>
<td>- amalgamate: 1</td>
</tr>
<tr>
<td>- 'node'</td>
</tr>
<tr>
<td>- 'at'</td>
</tr>
<tr>
<td>- 'to'</td>
</tr>
<tr>
<td>- 'decoration'</td>
</tr>
<tr>
<td>- '\plus'</td>
</tr>
<tr>
<td>- '\minus'</td>
</tr>
<tr>
<td>commandNameSpecial:</td>
</tr>
<tr>
<td>- amalgamate: 1</td>
</tr>
<tr>
<td>- '@ifnextchar['</td>
</tr>
</tbody>
</table>

roundParenthesesAllowed: 0|1

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 206.

<table>
<thead>
<tr>
<th>Listing 206: pstricks1.tex</th>
<th>Listing 207: pstricks1 default output</th>
</tr>
</thead>
<tbody>
<tr>
<td>\defFunction{algebraic}{torus}(u,v) {(2+\cos(u))*\cos(v+\Pi))|</td>
<td></td>
</tr>
<tr>
<td>{(2+\cos(u))*\sin(v+\Pi))|</td>
<td></td>
</tr>
<tr>
<td>{\sin(u))|</td>
<td></td>
</tr>
</tbody>
</table>

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 205, then latexindent.pl will allow round parenthesis between optional and mandatory arguments. In the case of the code in Listing 206, latexindent.pl finds all the arguments of \defFunction, both before and after \((u,v)\).

The default output from running latexindent.pl on Listing 206 actually leaves it unchanged (see Listing 207); note in particular, this is because of noAdditionalIndentGlobal as discussed on page 53.

Upon using the YAML settings in Listing 209, and running the command

```
cmh:\$ latexindent.pl pstricks1.tex -l noRoundParentheses.yaml
```

we obtain the output given in Listing 208.

<table>
<thead>
<tr>
<th>Listing 208: pstricks1.tex using Listing 209</th>
</tr>
</thead>
<tbody>
<tr>
<td>\defFunction{algebraic}{torus}(u,v) {(2+\cos(u))*\cos(v+\Pi))|</td>
</tr>
<tr>
<td>{(2+\cos(u))*\sin(v+\Pi))|</td>
</tr>
<tr>
<td>{\sin(u))|</td>
</tr>
</tbody>
</table>

Notice the difference between Listing 207 and Listing 208; in particular, in Listing 208, 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 44) which, by default, assume indentation for their body, and hence the tabbed indentation in Listing 208.

Let’s explore this using the YAML given in Listing 211 and run the command
then the output is as in Listing 210.

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

Notice in Listing 210 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 211.

```
stringsAllowedBetweenArguments: (fields)
```

tikz users may well specify code such as that given in Listing 212; processing this code using latexindent.pl gives the default output in Listing 213.

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

With reference to Listing 205 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 212, 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 215 and running the command

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

we receive the output given in Listing 214.
Notice that each line after the `\draw` command (its `body`) in Listing 214 has been given the appropriate two-spaces worth of indentation specified in Listing 215.

Let’s compare this with the output from using the YAML settings in Listing 217, and running the command

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

given in Listing 216.

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 44) 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 205 on page 61, 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 218 or Listing 219 is equivalent to using the settings in Listing 220.

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 221 means that only the strings specified in that field will be used.
5.5 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 219 to 221.

We may explore this feature further with the code in Listing 222, whose default output is given in Listing 223.

Let's compare this with the output from using the YAML settings in Listing 225, and running the command

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

given in Listing 224.

You might like to compare the output given in Listing 223 and Listing 224. Note, in particular, in Listing 223 that the foreach command has not included any of the subsequent strings, and that the braces have been treated as a namedGroupingBracesBrackets. In Listing 224 the foreach command has been allowed to have \x/\y and in between arguments because of the settings given in Listing 225.

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 226, which has default output in Listing 227.

Notice that in Listing 227 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 107.
5.5 Commands and the strings between their arguments

For demonstration, we can compare this output with that given in Listing 228 in which the settings from Listing 229 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.

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

LISTING 228: ifnextchar.tex using Listing 229

```
commandCodeBlocks:
  commandNameSpecial: 0
```

LISTING 229: no-ifnextchar.yaml

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 219 to 221.
SECTION 6

The -m (modifylinebreaks) switch

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

6.1 textWrapOptions: modifying line breaks by text wrapping ........................................ 67
   6.1.1 text wrapping on a per-code-block basis ...................................................... 70
   6.1.2 Summary of text wrapping .......................................................... 76
6.2 oneSentencePerLine: modifying line breaks for sentences ............................ 76
   6.2.1 sentencesFollow .......................................................... 78
   6.2.2 sentencesBeginWith ...................................................... 79
   6.2.3 sentencesEndWith .......................................................... 80
   6.2.4 Features of the oneSentencePerLine routine ........................................... 82
   6.2.5 text wrapping and indenting sentences ............................................... 83
6.3 removeParagraphLineBreaks: modifying line breaks for paragraphs ............ 85
6.4 Combining removeParagraphLineBreaks and textWrapOptions ................. 91
6.5 Poly-switches ........................................................ 92
6.6 modifyLineBreaks for environments ...................................................... 93
   6.6.1 Adding line breaks: BeginStartsOnOwnLine and BodyStartsOnOwnLine ...... 93
   6.6.2 Adding line breaks using EndStartsOnOwnLine and EndFinishesWithLineBreak 95
   6.6.3 poly-switches 1, 2, and 3 only add line breaks when necessary .......... 96
   6.6.4 Removing line breaks (poly-switches set to −1) ...................................... 97
   6.6.5 About trailing horizontal space .............................................. 98
   6.6.6 poly-switch line break removal and blank lines .................................. 99
6.7 Poly-switches for double back slash ...................................................... 100
   6.7.1 Double back slash starts on own line ........................................ 100
   6.7.2 Double back slash finishes with line break ........................................ 101
   6.7.3 Double back slash poly switches for specialBeginEnd ..................... 101
   6.7.4 Double back slash poly switches for optional and mandatory arguments ........ 102
   6.7.5 Double back slash optional square brackets .................................. 103
6.8 Poly-switches for other code blocks ...................................................... 103
As of Version 3.0, \texttt{latexindent.pl} has the \texttt{-m} switch, which permits \texttt{latexindent.pl} to modify line breaks, according to the specifications in the \texttt{modifyLineBreaks} field. 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 230.

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

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

\begin{verbatim}
\begin{tabular}{ll}
\hline
\textbf{preserveBlankLines: 0|1} \\
\hline
\end{tabular}
\end{verbatim}

This field is directly related to \texttt{poly-switches}, discussed below. 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.

\begin{verbatim}
\begin{tabular}{ll}
\hline
\textbf{condenseMultipleBlankLinesInto: (positive integer)} \\
\hline
\end{tabular}
\end{verbatim}

Assuming that this switch takes an integer value greater than 0, \texttt{latexindent.pl} will condense multiple blank lines into the number of blank lines illustrated by this switch. As an example, Listing 231 shows a sample file with blank lines; upon running

\begin{verbatim}
\texttt{cmh:}\$	exttt{latexindent.pl myfile.tex -m}
\end{verbatim}

the output is shown in Listing 232; 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}
\begin{tabular}{ll}
\hline
\texttt{LISTING 231: \texttt{mlb1.tex}} \\
before blank line \\
\hline
after blank line \\
\hline
after blank line \\
\hline
\end{tabular}
\end{verbatim}

\begin{verbatim}
\begin{tabular}{ll}
\hline
\texttt{LISTING 232: \texttt{mlb1.tex out output}} \\
before blank line \\
\hline
after blank line \\
\hline
after blank line \\
\hline
\end{tabular}
\end{verbatim}

6.1 \texttt{textWrapOptions: modifying line breaks by text wrapping}

When the \texttt{-m} switch is active \texttt{latexindent.pl} has the ability to wrap text using the options specified in the \texttt{textWrapOptions} field, see Listing 233. The value of \texttt{columns} specifies the column at which the text should be wrapped. By default, the value of \texttt{columns} is 0, so \texttt{latexindent.pl} will \textit{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.

\begin{verbatim}
\begin{tabular}{ll}
\hline
\texttt{LISTING 233: \texttt{textWrapOptions}} \\
\texttt{columns: 0} \\
\hline
\end{tabular}
\end{verbatim}
For example, consider the file given in Listing 234.

**Listing 234: textwrap1.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.

Using the file `textwrap1.yaml` in Listing 236, and running the command

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

we obtain the output in Listing 235.

**Listing 235: 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 237,

**Listing 237: 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 236,

```
cmh:~$ latexindent.pl -m textwrap2.tex -o textwrap2-mod1.tex -l textwrap1.yaml
```

then the output is as in Listing 238.
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 239:

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

% text wrapping does not apply to comments by latexindent.pl

and running the following command and continuing to use textwrap1.yaml from Listing 236,

```bash
cmh:~$ latexindent.pl -m textwrap3.tex -o textwrap3-mod1.tex -l textwrap1.yaml
```
then the output is as in Listing 240.

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

% text wrapping does not apply to comments by latexindent.pl

The text wrapping routine of latexindent.pl is performed by the Text::Wrap module, which provides a separator feature to separate lines with characters other than a new line (see [17]). By default, the separator is empty which means that a new line token will be used, but you can change it as you see fit.

For example starting with the file in Listing 241:

```
Here is a line of text.
```

and using textwrap2.yaml from Listing 243 with the following command

```bash
cmh:~$ latexindent.pl -m textwrap4.tex -o textwrap4-mod2.tex -l textwrap2.yaml
```
then we obtain the output in Listing 242.
There are options to specify the huge option for the Text::Wrap module [17]. This can be helpful if you would like to forbid the Text::Wrap routine from breaking words. For example, using the settings in Listings 245 and 247 and running the commands

cmh:~$ latexindent.pl -m textwrap4.tex -o=textwrap2B.tex -l textwrap2A.yaml


gives the respective output in Listings 244 and 246. You can also specify break in your settings, but I haven't found a useful reason to do this; see [17] for more details.

### 6.1.1 text wrapping on a per-code-block basis

By default, if the value of columns is greater than 0 and the \texttt{-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 \texttt{latexindent.pl} to apply \texttt{textWrap} on a per-code-block basis.

Thanks to [20] for their help in testing and shaping this feature.

The full details of \texttt{textWrapOptions} are shown in Listing 248. In particular, note the field \texttt{perCodeBlockBasis}: 0.

<table>
<thead>
<tr>
<th>Listing 242: textwrap4-mod2.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Here is a line of text.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 243: textwrap2.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>modifyLineBreaks:</td>
</tr>
<tr>
<td>textWrapOptions:</td>
</tr>
<tr>
<td>columns: 5</td>
</tr>
<tr>
<td>separator: &quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 244: textwrap4-mod2A.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Here is a line of text.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 245: textwrap2A.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>modifyLineBreaks:</td>
</tr>
<tr>
<td>textWrapOptions:</td>
</tr>
<tr>
<td>columns: 3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 246: textwrap4-mod2B.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Here is a line of text.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 247: textwrap2B.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>modifyLineBreaks:</td>
</tr>
<tr>
<td>textWrapOptions:</td>
</tr>
<tr>
<td>columns: 3</td>
</tr>
<tr>
<td>huge: overflow</td>
</tr>
</tbody>
</table>
6.1 textWrapOptions: modifying line breaks by text wrapping

The code blocks detailed in Listing 248 are with direct reference to those detailed in Table 1 on page 44. 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 309 on page 85; in fact, the two features can even be combined (this is detailed in Section 6.4 on page 91).

Let's explore these switches with reference to the code given in Listing 249; the text outside of the environment is considered part of the masterDocument.

Listing 249: textwrap5.tex

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

```latex
\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.

With reference to this code block, the settings given in Listings 250 to 252 each give the same output.

Listing 250: textwrap3.yaml

```yaml
modifyLineBreaks:
  textWrapOptions:
    columns: 30
    perCodeBlockBasis: 1
    all: 1
```

Listing 251: textwrap4.yaml

```yaml
modifyLineBreaks:
  textWrapOptions:
    columns: 30
    perCodeBlockBasis: 1
    environments: 1
    masterDocument: 1
```

Listing 252: textwrap5.yaml

```yaml
modifyLineBreaks:
  textWrapOptions:
    columns: 30
    perCodeBlockBasis: 1
    environments:
      myenv: 1
      masterDocument: 1
```

Let's explore the similarities and differences in the equivalent (with respect to Listing 249) syntax specified in Listings 250 to 252:

- in each of Listings 250 to 252 notice that columns: 30;
- in each of Listings 250 to 252 notice that perCodeBlockBasis: 1;
- in Listing 250 we have specified all: 1 so that the text wrapping will operate upon all code blocks;
- in Listing 251 we have not specified all, and instead, have specified that text wrapping should be applied to each of environments and masterDocument;
- in Listing 252 we have specified text wrapping for masterDocument and on a per-name basis.
6.1 textWrapOptions: modifying line breaks by text wrapping

for environments code blocks.

Upon running the following commands

\begin{verbatim}
cmh:~$ latexindent.pl -s textwrap5.tex -l=textwrap3.yaml -m
cmh:~$ latexindent.pl -s textwrap6.tex -l=textwrap4.yaml -m
cmh:~$ latexindent.pl -s textwrap5.tex -l=textwrap5.yaml -m
\end{verbatim}

we obtain the output shown in Listing 253.

\begin{center}
\textbf{Listing 253: textwrap5-mod3.tex}
\end{center}

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

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

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 252 by using Listing 254.

\begin{center}
\textbf{Listing 254: textwrap6.tex}
\end{center}

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

\begin{verbatim}
\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}
\end{verbatim}

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

In particular, upon running

\begin{verbatim}
cmh:~$ latexindent.pl -s textwrap6.tex -l=textwrap5.yaml -m
\end{verbatim}

we obtain the output given in Listing 255.
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 252) 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 254 in the settings given in Listings 256 to 258.

Upon running the commands

```bash
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 259 to 261.
6.1 textWrapOptions: modifying line breaks by text wrapping

<table>
<thead>
<tr>
<th>Listing 259: textwrap6.tex using Listing 256</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before the environment; here</td>
</tr>
<tr>
<td>is a line of text that can be</td>
</tr>
<tr>
<td>wrapped by latexindent.pl.</td>
</tr>
<tr>
<td>\begin{myenv}</td>
</tr>
<tr>
<td>Within the environment; here is a line</td>
</tr>
<tr>
<td>of text that can be wrapped by</td>
</tr>
<tr>
<td>latexindent.pl.</td>
</tr>
<tr>
<td>\end{myenv}</td>
</tr>
<tr>
<td>\begin{another}</td>
</tr>
<tr>
<td>Within the environment; here is a line</td>
</tr>
<tr>
<td>of text that can be wrapped by</td>
</tr>
<tr>
<td>latexindent.pl.</td>
</tr>
<tr>
<td>\end{another}</td>
</tr>
<tr>
<td>After the environment; here</td>
</tr>
<tr>
<td>is a line of text that can be</td>
</tr>
<tr>
<td>wrapped by latexindent.pl.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 260: textwrap6.tex using Listing 257</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before the environment; here</td>
</tr>
<tr>
<td>is a line of text that can be</td>
</tr>
<tr>
<td>wrapped by latexindent.pl.</td>
</tr>
<tr>
<td>\begin{myenv}</td>
</tr>
<tr>
<td>Within the environment; here is a line</td>
</tr>
<tr>
<td>of text that can be wrapped by</td>
</tr>
<tr>
<td>latexindent.pl.</td>
</tr>
<tr>
<td>\end{myenv}</td>
</tr>
<tr>
<td>\begin{another}</td>
</tr>
<tr>
<td>Within the environment; here is a line</td>
</tr>
<tr>
<td>of text that can be wrapped by</td>
</tr>
<tr>
<td>latexindent.pl.</td>
</tr>
<tr>
<td>\end{another}</td>
</tr>
<tr>
<td>After the environment; here</td>
</tr>
<tr>
<td>is a line of text that can be</td>
</tr>
<tr>
<td>wrapped by latexindent.pl.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 261: textwrap6.tex using Listing 258</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before the environment; here</td>
</tr>
<tr>
<td>is a line of text that can be</td>
</tr>
<tr>
<td>wrapped by latexindent.pl.</td>
</tr>
<tr>
<td>\begin{myenv}</td>
</tr>
<tr>
<td>Within the environment; here is a line</td>
</tr>
<tr>
<td>of text that can be wrapped by</td>
</tr>
<tr>
<td>latexindent.pl.</td>
</tr>
<tr>
<td>\end{myenv}</td>
</tr>
<tr>
<td>\begin{another}</td>
</tr>
<tr>
<td>Within the environment; here is a line</td>
</tr>
<tr>
<td>of text that can be wrapped by</td>
</tr>
<tr>
<td>latexindent.pl.</td>
</tr>
<tr>
<td>\end{another}</td>
</tr>
<tr>
<td>After the environment; here</td>
</tr>
<tr>
<td>is a line of text that can be</td>
</tr>
<tr>
<td>wrapped by latexindent.pl.</td>
</tr>
</tbody>
</table>

Notice that:

- in Listing 259 the text wrapping routine has not been applied to any environments because it has been switched off (per-code-block) in Listing 256;
• in Listing 260 the text wrapping routine has not been applied to myenv because it has been switched off (per-name) in Listing 257;

• in Listing 261 the text wrapping routine has not been applied to masterDocument because of the settings in Listing 258.

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 254 on page 72, for example). We explore further options in Listings 262 to 264.

Listing 262 and Listing 263 are equivalent. Upon running the commands

```bash
cmh:~$ latexindent.pl -s textwrap6.tex -l=textwrap9.yaml -m
cmh:~$ latexindent.pl -s textwrap6.tex -l=textwrap10.yaml -m
```

we receive the respective output given in Listings 265 and 266.

**Listing 262: textwrap9.yaml**

```yaml
modifyLineBreaks:
textWrapOptions:
columns:
  default: 30
  environments: 50
  perCodeBlockBasis: 1
  all: 1
```

**Listing 263: textwrap10.yaml**

```yaml
modifyLineBreaks:
textWrapOptions:
columns:
  default: 30
  environments:
    myenv: 50
    another: 15
  perCodeBlockBasis: 1
  all: 1
```

**Listing 264: textwrap11.yaml**

```yaml
modifyLineBreaks:
textWrapOptions:
columns:
  default: 30
  environments:
    myenv: 50
    another: 15
  perCodeBlockBasis: 1
  all: 1
```

Listing 265: textwrap6.tex using Listing 262

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

\begin{myenv}
\begin{verbatim}
Within the environment; here is a line of text that can be wrapped by latexindent.pl.
\end{verbatim}
\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 265 the text for the masterDocument has been wrapped using 30 columns, while environments has been wrapped using 50 columns;
- in Listing 266 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 26 on page 28 then text wrapping will be disabled for this code block.

If you wish to specify afterHeading commands (see Listing 98 on page 41) on a per-name basis, then you need to append the name with :heading, for example, you might use section:heading.

### 6.1.2 Summary of text wrapping

It is important to note the following:

- Verbatim environments (Listing 17 on page 26) and verbatim commands (Listing 18 on page 26) will not be affected by the text wrapping routine (see Listing 238 on page 69);
- comments will not be affected by the text wrapping routine (see Listing 240 on page 69);
- it is possible to wrap text on a per-code-block and a per-name basis;
- the text wrapping routine sets preserveBlankLines as 1;
- indentation is performed after the text wrapping routine; as such, indented code will likely exceed any maximum value set in the columns field.

### 6.2 oneSentencePerLine: modifying line breaks for sentences

You can instruct latexindent.pl to format your file so that it puts one sentence per line. Thank you to [12] for helping to shape and test this feature. The behaviour of this part of the script is controlled by the switches detailed in Listing 267, all of which we discuss next.
6.2 oneSentencePerLine: modifying line breaks for sentences

<table>
<thead>
<tr>
<th>LISTING 267: oneSentencePerLine</th>
</tr>
</thead>
<tbody>
<tr>
<td>448 oneSentencePerLine:</td>
</tr>
<tr>
<td>449 manipulateSentences: 0</td>
</tr>
<tr>
<td>450 removeSentenceLineBreaks: 1</td>
</tr>
<tr>
<td>451 textWrapSentences: 0</td>
</tr>
<tr>
<td>452 sentenceIndent: &quot;&quot;</td>
</tr>
<tr>
<td>453 sentencesFollow:</td>
</tr>
<tr>
<td>454 par: 1</td>
</tr>
<tr>
<td>455 blankLine: 1</td>
</tr>
<tr>
<td>456 fullStop: 1</td>
</tr>
<tr>
<td>457 exclamationMark: 1</td>
</tr>
<tr>
<td>458 questionMark: 1</td>
</tr>
<tr>
<td>459 rightBrace: 1</td>
</tr>
<tr>
<td>460 commentOnPreviousLine: 1</td>
</tr>
<tr>
<td>461 other: 0</td>
</tr>
<tr>
<td>462 sentencesBeginWith:</td>
</tr>
<tr>
<td>463 A-Z: 1</td>
</tr>
<tr>
<td>464 a-z: 0</td>
</tr>
<tr>
<td>465 other: 0</td>
</tr>
<tr>
<td>466 sentencesEndWith:</td>
</tr>
<tr>
<td>467 basicFullStop: 0</td>
</tr>
<tr>
<td>468 betterFullStop: 1</td>
</tr>
<tr>
<td>469 exclamationMark: 1</td>
</tr>
<tr>
<td>470 questionMark: 1</td>
</tr>
<tr>
<td>471 other: 0</td>
</tr>
</tbody>
</table>

**manipulateSentences**: 0|1

This is a binary switch that details if `latexindent.pl` should perform the sentence manipulation routine; it is off (set to 0) by default, and you will need to turn it on (by setting it to 1) if you want the script to modify line breaks surrounding and within sentences.

**removeSentenceLineBreaks**: 0|1

When operating upon sentences `latexindent.pl` will, by default, remove internal line breaks as `removeSentenceLineBreaks` is set to 1. Setting this switch to 0 instructs `latexindent.pl` not to do so.

For example, consider `multiple-sentences.tex` shown in Listing 268.

<table>
<thead>
<tr>
<th>LISTING 268: multiple-sentences.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>This is the first sentence. This is the; second, sentence. This is the third sentence.</td>
</tr>
<tr>
<td>This is the fourth sentence! This is the fifth sentence? This is the sixth sentence.</td>
</tr>
</tbody>
</table>

If we use the YAML files in Listings 270 and 272, and run the commands

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

then we obtain the respective output given in Listings 269 and 271.
6.2 oneSentencePerLine: modifying line breaks for sentences

Notice, in particular, that the ‘internal’ sentence line breaks in Listing 268 have been removed in Listing 269, but have not been removed in Listing 271.

The remainder of the settings displayed in Listing 267 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 273); 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 274); by default, this is only capital letters;
- end with a character (see Listing 275); by default, these are full stop/period (.), exclamation mark (!) and question mark (?).

In each case, you can specify the other field to include any pattern that you would like; you can specify anything in this field using the language of regular expressions.

6.2.1 sentencesFollow

Let’s explore a few of the switches in sentencesFollow; let’s start with Listing 268 on the previous page, and use the YAML settings given in Listing 277. Using the command

```
cmh:~$ latexindent.pl multiple-sentences -m -l=sentences-follow1.yaml
```
we obtain the output given in Listing 276.

**Listing 276: multiple-sentences.tex using Listing 277**

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 273 with the `.tex` file detailed in Listing 278.

**Listing 278: 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

```bash
$ latexindent.pl multiple-sentences1 -m -l=manipulate-sentences.yaml
$ latexindent.pl multiple-sentences1 -m -l=manipulate-sentences.yaml,sentences-follow2.yaml
```

then we obtain the respective output given in Listings 279 and 280.

**Listing 279: multiple-sentences1.tex using Listing 270 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 280: multiple-sentences1.tex using Listing 281**

(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 279 the first sentence after the `)` has not been accounted for, but that following the inclusion of Listing 281, the output given in Listing 280 demonstrates that the sentence has been accounted for correctly.

### 6.2.2 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 274), and we can use the other field to define sentences to begin with other characters.
6.2 oneSentencePerLine: modifying line breaks for sentences

\begin{Verbatim}
LISTING 282: multiple-sentences2.tex
\end{Verbatim}

This is the first sentence.

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

Upon running the following commands

\begin{Verbatim}
cmh:\$\sim\$ latexindent.pl multiple-sentences2 \texttt{-m -l=manipulate-sentences.yaml}
cmh:\$\sim\$ latexindent.pl multiple-sentences2 \texttt{-m -l=manipulate-sentences.yaml,sentences-begin1.yaml}
\end{Verbatim}

then we obtain the respective output given in Listings 283 and 284.

\begin{Verbatim}
LISTING 283: multiple-sentences2.tex using Listing 270 on page 78
\end{Verbatim}

This is the first sentence.

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

\begin{Verbatim}
LISTING 284: multiple-sentences2.tex using Listing 285
\end{Verbatim}

This is the first sentence.

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

Notice that in Listing 283, the first sentence has been accounted for but that the subsequent sentences have not. In Listing 284, all of the sentences have been accounted for, because the other field in Listing 285 has defined sentences to begin with either \$ or any numeric digit, 0 to 9.

6.2.3 sentencesEndWith

Let’s return to Listing 268 on page 77; we have already seen the default way in which latexindent.pl will operate on the sentences in this file in Listing 269 on page 78. We can populate the other field with any character that we wish; for example, using the YAML specified in Listing 287 and the command

\begin{Verbatim}
cmh:\$\sim\$ latexindent.pl multiple-sentences -m -l=sentences-end1.yaml
cmh:\$\sim\$ latexindent.pl multiple-sentences -m -l=sentences-end2.yaml
\end{Verbatim}

then we obtain the output in Listing 286.

\begin{Verbatim}
LISTING 286: multiple-sentences.tex using Listing 287
\end{Verbatim}

This is the first sentence.
This is the second sentence.
This is the third sentence.
This is the fourth sentence!
This is the fifth sentence?
This is the sixth sentence.

\begin{Verbatim}
LISTING 287: sentences-end1.yaml
\end{Verbatim}
6.2 oneSentencePerLine: modifying line breaks for sentences

There is a subtle difference between the output in Listings 286 and 288; in particular, in Listing 286 the word sentence has not been defined as a sentence, because we have not instructed \texttt{latexindent.pl} to begin sentences with lower case letters. We have changed this by using the settings in Listing 289, and the associated output in Listing 288 reflects this.

Referencing Listing 275 on page 78, you’ll notice that there is a field called \texttt{basicFullStop}, which is set to 0, and that the \texttt{betterFullStop} is set to 1 by default.

Let’s consider the file shown in Listing 290.

Upon running the following commands

\texttt{cmh:~$ latexindent.pl url \textbackslash-m \textbackslash-l=manipulate-sentences.yaml}

we obtain the output given in Listing 291.

Notice that the full stop within the url has been interpreted correctly. This is because, within the \texttt{betterFullStop}, full stops at the end of sentences have the following properties:

- they are ignored within \texttt{e.g.} and \texttt{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 \texttt{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 \texttt{betterFullStop} routine by using the \texttt{fine tuning}, detailed in Listing 470 on page 118.

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

\texttt{cmh:~$ latexindent.pl url \textbackslash-m \textbackslash-l=alt-full-stop1.yaml}

and the YAML in Listing 293 gives the output in Listing 292.
Notice that the full stop within the URL has not been accommodated correctly because of the non-default settings in Listing 293.

### 6.2.4 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 294, and run the command

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

then we obtain the output in Listing 295.

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 296 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 297 and 298.
Once you've read Section 6.5, you will know that you can accommodate the removal of internal sentence line breaks by using the YAML in Listing 300 and the command

```bash
$ latexindent pl multiple-sentences4 -m -l=item-rules2.yaml
```

the output of which is shown in Listing 299.

### 6.2.5 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 301.

Referencing Listing 303, and running the following command

```bash
$ latexindent.pl multiple-sentences5 -m -l=sentence-wrap1.yaml
```
we receive the output given in Listing 302.

If you wish to specify the columns field on a per-code-block basis for sentences, then you would use sentence; explicitly, starting with Listing 262 on page 75, for example, you would replace/append environments with, for example, sentence: 50.

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

The indentation of sentences requires that sentences are stored as code blocks. This means that you may need to tweak Listing 275 on page 78. Let’s explore this in relation to Listing 304.

Consider the following:
\begin{itemize}
\item firstly.
\item secondly.
\end{itemize}

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

\begin{verbatim}
cmh:~$ latexindent.pl multiple-sentences6 -m -l=sentence-wrap1.yaml
\end{verbatim}

we receive the respective output in Listing 305 and Listing 306.

We note that Listing 305 the itemize code block has not been indented appropriately. This is because the oneSentencePerLine has been instructed to store sentences (because Listing 303); each sentence is then searched for code blocks.

We can tweak the settings in Listing 275 on page 78 to ensure that full stops are not followed by item commands, and that the end of sentences contains \end{itemize} as in Listing 307 (if you intend to
6.3 removeParagraphLineBreaks: modifying line breaks for paragraphs

Upon running

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

we receive the output in Listing 308.

Consider the following:

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

Notice that the sentence has received indentation, and that the `itemize` code block has been found and indented correctly.

6.3 removeParagraphLineBreaks: modifying line breaks for paragraphs

When the `-m` switch is active `latexindent.pl` has the ability to remove line breaks from within paragraphs; the behaviour is controlled by the `removeParagraphLineBreaks` field, detailed in Listing 309. Thank you to [13] for shaping and assisting with the testing of this feature.

This feature is considered complimentary to the `oneSentencePerLine` feature described in Section 6.2 on page 76.

This routine can be turned on `globally` for every code block type known to `latexindent.pl` (see Table 1 on page 44) by using the `all` switch; by default, this switch is `off`. Assuming that the `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 `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 44;

• it can include line breaks, but finishes when it meets either a blank line, a `\par` command, or any of the user-specified settings in the `paragraphsStopAt` field, detailed in Listing 326 on page 90.

Let’s start with the `.tex` file in Listing 310, together with the YAML settings in Listing 311.

<table>
<thead>
<tr>
<th>Listing 310: shortlines.tex</th>
<th>Listing 311: remove-para1.yaml</th>
</tr>
</thead>
</table>
| `\begin{myenv}
The lines in this environment are very short and contain many linebreaks.

Another paragraph.
\end{myenv}` | `modifyLineBreaks:
removeParagraphLineBreaks:
  all: 1` |

Upon running the command

```
cmh:~$ latexindent.pl -m shortlines.tex -o shortlines1.tex -l remove-para1.yaml
```

then we obtain the output given in Listing 312.

<table>
<thead>
<tr>
<th>Listing 312: shortlines1.tex</th>
</tr>
</thead>
</table>
| `\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 310 which has crept in to the output in Listing 312. This can be fixed using the YAML file in Listing 386 on page 98 and running, for example,

```
cmh:~$ latexindent.pl -m shortlines.tex -o shortlines1-tws.tex -l remove-para1.yaml,removeTWS-before.yaml
```

in which case the output is as in Listing 313; notice that the double spaces present in Listing 312 have been addressed.

<table>
<thead>
<tr>
<th>Listing 313: shortlines1-tws.tex</th>
</tr>
</thead>
</table>
| `\begin{myenv}
The lines in this environment are very short and contain many linebreaks.

Another paragraph.
\end{myenv}` |

Keeping with the settings in Listing 311, we note that the all switch applies to all code block types. So, for example, let’s consider the files in Listings 314 and 315.
6.3 removeParagraphLineBreaks: modifying line breaks for paragraphs

<table>
<thead>
<tr>
<th>Listing 314: shortlines-mand.tex</th>
<th>Listing 315: shortlines-opt.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>\mycommand{ \ The lines in this command are very short and contain many linebreaks. \ } \ Another paragraph. \ }</td>
<td>\mycommand[ The lines in this command are very short and contain many linebreaks. \ ] \ Another paragraph. \ ]</td>
</tr>
</tbody>
</table>

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 316 and 317.

<table>
<thead>
<tr>
<th>Listing 316: shortlines-mand1.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>\mycommand{ \ The lines in this command are very short and contain many linebreaks. \ } \ Another paragraph. \ }</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 317: shortlines-opt1.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>\mycommand[ The lines in this command are very short and contain many linebreaks. \ ] \ Another paragraph. \ ]</td>
</tr>
</tbody>
</table>

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

For example, let’s use the code in Listing 318, and consider the settings in Listings 319 and 320; note that in Listing 319 we specify that every environment should receive treatment from the routine, while in Listing 320 we specify that only the one environment should receive the treatment.
6.3 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}

Upon running the commands

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

then we obtain the respective output given in Listings 321 and 322.

\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}
The lines in this environment are very short and contain many line breaks.

Another paragraph.

The lines in this environment are very short and contain many line breaks.

Another paragraph.

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 323, with the YAML settings in Listing 324.

Upon running the following command

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

then we obtain the output in Listing 325.
The lines in this document are very short and contain many linebreaks.

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

Another paragraph.

\begin{myenv}
  Body of myenv
\end{myenv}

Another paragraph.

\begin{myenv}
  Body of myenv
\end{myenv}

Note that the all field can take the same exceptions detailed in Listing 256lst:textwrap8-yaml.

The paragraph line break routine considers blank lines and the \par command to be the end of a paragraph; you can tune the behaviour of the routine further by using the paragraphsStopAt fields, shown in Listing 326.

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 326. 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 327; we will, in turn, consider the settings in Listings 328 and 329.

Upon using the settings from Listing 324 on the preceding page and running the commands
we obtain the respective outputs in Listings 330 to 332; notice in particular that:

- in Listing 330 the paragraph line break routine has included commands and comments;
- in Listing 331 the paragraph line break routine has stopped at the emph command, because in Listing 328 we have specified commands to be 1, and emph is at the beginning of a line;
- in Listing 332 the paragraph line break routine has stopped at the comments, because in Listing 329 we have specified comments to be 1, and the comment is at the beginning of a line.

In all outputs in Listings 330 to 332 we notice that the paragraph line break routine has stopped at \begin{myenv} because, by default, environments is set to 1 in Listing 326 on the previous page.

---

**LISTING 330: sl-stop4.tex**

These lines are very short \textit{and} contain many linebreaks.
\begin{myenv}
Body of myenv
\end{myenv}

Another paragraph. \% a comment\% a comment

**LISTING 331: sl-stop4-command.tex**

These lines are very short \textit{and} contain many linebreaks.
\begin{myenv}
Body of myenv
\end{myenv}

Another paragraph. \% a comment\% a comment

**LISTING 332: sl-stop4-comment.tex**

These lines are very short \textit{and} contain many linebreaks.
\begin{myenv}
Body of myenv
\end{myenv}

Another paragraph.
\% a comment
\% a comment

---

**6.4 Combining removeParagraphLineBreaks and textWrapOptions**

The text wrapping routine (Section 6.1 on page 67) and remove paragraph line breaks routine (Section 6.3 on page 85) can be combined.

We motivate this feature with the code given in Listing 333.

**LISTING 333: textwrap7.tex**

This paragraph has line breaks throughout its paragraph; we would like to combine the textwrapping and paragraph removal routine.
Applying the text wrap routine from Section 6.1 on page 67 with, for example, Listing 250 on page 71 gives the output in Listing 334.

<table>
<thead>
<tr>
<th>LISTING 334: textwrap7.tex using Listing 250</th>
</tr>
</thead>
<tbody>
<tr>
<td>This paragraph has line breaks throughout its paragraph; we would like to combine the text wrapping and paragraph removal routine.</td>
</tr>
</tbody>
</table>

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 336 and running the command

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

we obtain the output in Listing 335.

<table>
<thead>
<tr>
<th>LISTING 335: textwrap7-mod12.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>This paragraph has line breaks throughout its paragraph; we would like to combine the text wrapping and paragraph removal routine.</td>
</tr>
</tbody>
</table>

In Listing 335 the paragraph line breaks have first been removed from Listing 333, and then the text wrapping routine has been applied. It is envisaged that variants of Listing 336 will be among the most useful settings for these two features.

### 6.5 Poly-switches

Every other field in the modifyLineBreaks field uses poly-switches, and can take one of five integer values:

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

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

- 1 **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 **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 **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 **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 `begin statement`, `body` or `end statement` of the code blocks detailed in Table 1 on page 44. All poly-switches are off by default; `latexindent.pl` searches first of all for per-name settings, and then followed by global per-thing settings.
6.6 modifyLineBreaks for environments

We start by viewing a snippet of defaultSettings.yaml in Listing 337; note that it contains global settings (immediately after the environments field) and that per-name settings are also allowed – in the case of Listing 337, settings for equation* have been specified for demonstration. Note that all poly-switches are off (set to 0) by default.

<table>
<thead>
<tr>
<th>Listing 337: environments</th>
</tr>
</thead>
<tbody>
<tr>
<td>environments</td>
</tr>
<tr>
<td>BeginStartsOnOwnLine: 0</td>
</tr>
<tr>
<td>BodyStartsOnOwnLine: 0</td>
</tr>
<tr>
<td>EndStartsOnOwnLine: 0</td>
</tr>
<tr>
<td>EndFinishesWithLineBreak: 0</td>
</tr>
<tr>
<td>equation*:</td>
</tr>
<tr>
<td>BeginStartsOnOwnLine: 0</td>
</tr>
<tr>
<td>BodyStartsOnOwnLine: 0</td>
</tr>
<tr>
<td>EndStartsOnOwnLine: 0</td>
</tr>
<tr>
<td>EndFinishesWithLineBreak: 0</td>
</tr>
</tbody>
</table>

Let’s begin with the simple example given in Listing 338; note that we have annotated key parts of the file using ♠, ♥, ♦ and ♣, these will be related to fields specified in Listing 337.

Listing 338: env-mlb1.tex

before words♠ \begin{myenv}♥ body of myenv♦ \end{myenv}♣ after words

6.6.1 Adding line breaks: BeginStartsOnOwnLine and BodyStartsOnOwnLine

Let’s explore BeginStartsOnOwnLine and BodyStartsOnOwnLine in Listings 339 and 340, and in particular, let’s allow each of them in turn to take a value of 1.

Listing 339: env-mlb1.yaml

modifyLineBreaks:
environments:
BeginStartsOnOwnLine: 1

Listing 340: env-mlb2.yaml

modifyLineBreaks:
environments:
BodyStartsOnOwnLine: 1

After running the following commands,

cmh:~$ latexindent -m env-mlb1.pi -l env-mlb1.yaml

cmh:~$ latexindent -m env-mlb2.pi -l env-mlb2.yaml

the output is as in Listings 341 and 342 respectively.

Listing 341: env-mlb.tex using Listing 339

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

Listing 342: env-mlb.tex using Listing 340

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

There are a couple of points to note:

- in Listing 341 a line break has been added at the point denoted by ♠ in Listing 338; no other line breaks have been changed;
- in Listing 342 a line break has been added at the point denoted by ♥ in Listing 338; furthermore, note that the body of myenv has received the appropriate (default) indentation.

Let’s now change each of the 1 values in Listings 339 and 340 so that they are 2 and save them into env-mlb3.yaml and env-mlb4.yaml respectively (see Listings 343 and 344).

Listing 343: env-mlb3.yaml

modifyLineBreaks:
environments:
BeginStartsOnOwnLine: 2

Listing 344: env-mlb4.yaml

modifyLineBreaks:
environments:
BodyStartsOnOwnLine: 2
Upon running commands analogous to the above, we obtain Listings 345 and 346.

Upon running commands analogous to the above, we obtain Listings 349 and 350.

Let's now change each of the \( n \) values in Listings 339 and 340 so that they are 3 and save them into env-mlb5.yaml and env-mlb6.yaml respectively (see Listings 347 and 348).

Upon running commands analogous to the above, we obtain Listings 349 and 350.

We will demonstrate this poly-switch value using the code in Listing 353.

Upon running the commands

\[
\text{cmh:}\sim\$\text{ latexindent.pl }-m\text{ env-mlb1.tex -l env-beg4.yaml}
\text{cmh:}\sim\$\text{ latexindent.pl }-m\text{ env-mlb1.tex -l env-body4.yaml}
\]

then we receive the respective outputs in Listings 354 and 355.
We note in particular that, by design, for this value of the poly-switches:

1. in Listing 354 a blank line has been inserted before the \texttt{\begin} statement, even though the \texttt{\begin} statement was already on its own line;
2. in Listing 355 a blank line has been inserted before the beginning of the \texttt{body}, even though it already began on its own line.

### 6.6.2 Adding line breaks using \texttt{EndStartsOnOwnLine} and \texttt{EndFinishesWithLineBreak}

Let's explore \texttt{EndStartsOnOwnLine} and \texttt{EndFinishesWithLineBreak} in Listings 356 and 357, and in particular, let's allow each of them in turn to take a value of 1.

**Listing 356: env-mlb7.yaml**

```yaml
modifyLineBreaks:
  environments:
    EndStartsOnOwnLine: 1
```

**Listing 357: env-mlb8.yaml**

```yaml
modifyLineBreaks:
  environments:
    EndFinishesWithLineBreak: 1
```

After running the following commands,

```
cmh:∼$ latexindent -m env-mlb.tex -l env-mlb7.yaml
cmh:∼$ latexindent -m env-mlb.tex -l env-mlb8.yaml
```

the output is as in Listings 358 and 359.

**Listing 358: env-mlb.tex using Listing 356**

```latex
before words \texttt{\begin}body of myenv \texttt{\end} after words
```

**Listing 359: env-mlb.tex using Listing 357**

```latex
before words \texttt{\begin}body of myenv\texttt{\end} after words
```

There are a couple of points to note:

- in Listing 358 a line break has been added at the point denoted by ♦ in Listing 338 on page 93; no other line breaks have been changed and the \texttt{\end} statement has not received indentation (as intended);
- in Listing 359 a line break has been added at the point denoted by ♦ in Listing 338 on page 93.

Let's now change each of the 1 values in Listings 356 and 357 so that they are 2 and save them into env-mlb9.yaml and env-mlb10.yaml respectively (see Listings 360 and 361).

**Listing 360: env-mlb9.yaml**

```yaml
modifyLineBreaks:
  environments:
    EndStartsOnOwnLine: 2
```

**Listing 361: env-mlb10.yaml**

```yaml
modifyLineBreaks:
  environments:
    EndFinishesWithLineBreak: 2
```

Upon running commands analogous to the above, we obtain Listings 362 and 363.

**Listing 362: env-mlb.tex using Listing 360**

```latex
before words \texttt{\begin}body of myenv \texttt{\end} after words
```

**Listing 363: env-mlb.tex using Listing 361**

```latex
before words \texttt{\begin}body of myenv\texttt{\end} after words
```

Note that line breaks have been added as in Listings 358 and 359, 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 356 and 357 so that they are 3 and save them into env-mlb11.yaml and env-mlb12.yaml respectively (see Listings 364 and 365).

**Listing 364: env-mlb11.yaml**

```yaml
modifyLineBreaks:
  environments:
    EndStartsOnOwnLine: 3
```

**Listing 365: env-mlb12.yaml**

```yaml
modifyLineBreaks:
  environments:
    EndFinishesWithLineBreak: 3
```
Upon running commands analogous to the above, we obtain Listings 366 and 367.

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>before words \begin{myenv}body of myenv</td>
<td>before words \begin{myenv}body of myenv\end{myenv}</td>
</tr>
<tr>
<td>\end{myenv} after words</td>
<td>after words</td>
</tr>
</tbody>
</table>

Note that line breaks have been added as in Listings 358 and 359, and that a blank line has been added after the line break.

Let's now change each of the 1 values in Listings 364 and 365 so that they are 4 and save them into env-end4.yaml and env-end-f4.yaml respectively (see Listings 368 and 369).

<table>
<thead>
<tr>
<th>LISTING 368: env-end4.yaml</th>
<th>LISTING 369: env-end-f4.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>modifyLineBreaks:</td>
<td>modifyLineBreaks:</td>
</tr>
<tr>
<td>environments:</td>
<td>environments:</td>
</tr>
<tr>
<td>EndStartsOnOwnLine: 4</td>
<td>EndFinishesWithLineBreak: 4</td>
</tr>
</tbody>
</table>

We will demonstrate this poly-switch value using the code from Listing 353 on page 94.

Upon running the commands

```
cmh:~$ latexindent.pl -m env-mlb1.tex -l env-end4.yaml
cmh:~$ latexindent.pl -m env-mlb1.tex -l env-end-f4.yaml
```

then we receive the respective outputs in Listings 370 and 371.

<table>
<thead>
<tr>
<th>LISTING 370: env-mlb1.tex using Listing 368</th>
<th>LISTING 371: env-mlb1.tex using Listing 369</th>
</tr>
</thead>
<tbody>
<tr>
<td>before words \begin{myenv}body of myenv</td>
<td>before words \begin{myenv}body of myenv\end{myenv}</td>
</tr>
<tr>
<td>\end{myenv} after words</td>
<td>after words</td>
</tr>
</tbody>
</table>

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

1. in Listing 370 a blank line has been inserted before the \end statement, even though the \end statement was already on its own line;
2. in Listing 371 a blank line has been inserted after the \end statement, even though it already began on its own line.

### 6.6.3 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 372 using poly-switch values of 1, 2, or 3, it will be left unchanged.

<table>
<thead>
<tr>
<th>LISTING 372: env-mlb2.tex</th>
<th>LISTING 373: env-mlb3.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>before words \begin{myenv}body of myenv\end{myenv}</td>
<td>before words \begin{myenv}body of myenv% \end{myenv}%</td>
</tr>
<tr>
<td>after words</td>
<td>after words</td>
</tr>
</tbody>
</table>

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 354 and 355 and Listings 370 and 371.

In contrast, the output from processing the file in Listing 373 will vary depending on the poly-switches used; in Listing 374 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 375 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 373 and by setting the other poly-switches considered so far to 2 in turn.

### Listing 374: env-mlb3.tex using Listing 340 on page 93

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

after words

### Listing 375: env-mlb3.tex using Listing 344 on page 93

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

after words

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 337 on page 93, an example is shown for the equation* environment.

### 6.6.4 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 376, noting in particular the positions of the line break highlighters, ♠, ♥, ♦ and ♣, together with the associated YAML files in Listings 377 to 380.

### Listing 376: env-mlb4.tex

before words♠
\begin{myenv}
  ♥
  body of myenv♥
\end{myenv}♣

after words

### Listing 377: env-mlb13.yaml

modifyLineBreaks:
environments:
  BeginStartsOnOwnLine: -1

### Listing 378: env-mlb14.yaml

modifyLineBreaks:
environments:
  BodyStartsOnOwnLine: -1

### Listing 379: env-mlb15.yaml

modifyLineBreaks:
environments:
  EndStartsOnOwnLine: -1

### Listing 380: env-mlb16.yaml

modifyLineBreaks:
environments:
  EndFinishesWithLineBreak: -1

After running the commands

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

we obtain the respective output in Listings 381 to 384.
Notice that in:

- Listing 381 the line break denoted by ♠ in Listing 376 has been removed;
- Listing 382 the line break denoted by ♥ in Listing 376 has been removed;
- Listing 383 the line break denoted by ♦ in Listing 376 has been removed;
- Listing 384 the line break denoted by ♣ in Listing 376 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 377 to 380 into one file; alternatively, you could tell \texttt{latexindent.pl} to load them all by using the following command, for example

\begin{verbatim}
\end{verbatim}

which gives the output in Listing 338 on page 93.

### 6.6.5 About trailing horizontal space

Recall that on page 26 we discussed the YAML field \texttt{removeTrailingWhitespace}, and that it has two (binary) switches to determine if horizontal space should be removed \texttt{beforeProcessing} and \texttt{afterProcessing}. The \texttt{beforeProcessing} is particularly relevant when considering the \texttt{-m} switch; let's consider the file shown in Listing 385, which highlights trailing spaces.

\begin{verbatim}
Listing 385: env-mlb5.tex
before_words♠
\begin{myenv}♥
body_of_myenv♦
\end{myenv}♣
after_words
\end{verbatim}

The output from the following commands

\begin{verbatim}

Listing 385: env-mlb5.tex
before_words♠
\begin{myenv}♥
body_of_myenv♦
\end{myenv}♣
after_words
\end{verbatim}

\begin{verbatim}
cmh:~$ latexindent.pl -m env-mlb5.tex -l env-mlb13.yaml,env-mlb14.yaml,env-mlb15.yaml,env-mlb16.yaml,removeTWS-before.yaml

Listing 386: removeTWS-before.yaml
removeTrailingWhitespace:
beforeProcessing: 1
\end{verbatim}

is shown, respectively, in Listings 387 and 388; note that the trailing horizontal white space has been preserved (by default) in Listing 387, while in Listing 388, it has been removed using the switch specified in Listing 386.
6.6.6 poly-switch line break removal and blank lines

Now let's consider the file in Listing 389, which contains blank lines.

Listing 389: env-mlb6.tex

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

Upon running the following commands

```
```

we receive the respective outputs in Listings 391 and 392. In Listing 391 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 392, we have allowed the poly-switches to remove blank lines because, in Listing 390, we have set preserveBlankLines to 0.

Listing 390: UnpreserveBlankLines.yaml

```
modifyLineBreaks:
  preserveBlankLines: 0
```

We can explore this further using the blank-line poly-switch value of 3; let's use the file given in Listing 393.

Listing 393: env-mlb7.tex

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

Upon running the following commands

```
$ latexindent.pl -m env-mlb7.tex -l env-mlb12.yaml,env-mlb13.yaml
```

we receive the outputs given in Listings 394 and 395.
Notice that in:

- Listing 394 that \end{one} has added a blank line, because of the value of EndFinishesWithLineBreak in Listing 365 on page 95, and even though the line break ahead of \begin{two} should have been removed (because of BeginStartsOnOwnLine in Listing 377 on page 97), the blank line has been preserved by default;
- Listing 395, by contrast, has had the additional line-break removed, because of the settings in Listing 390.

### 6.7 Poly-switches for double back slash

With reference to lookForAlignDelims (see Listing 26 on page 28) you can specify poly-switches to dictate the line-break behaviour of double back slashes in environments (Listing 28 on page 28), commands (Listing 50 on page 33), or special code blocks (Listing 85 on page 38). Note that for these poly-switches to take effect, the name of the code block must necessarily be specified within lookForAlignDelims (Listing 26 on page 28); we will demonstrate this in what follows.

Consider the code given in Listing 396.

### Listing 396: tabular3.tex

```latex
\begin{tabular}{cc}
1 & 2
\end{tabular}
```

Referencing Listing 396:

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

Let's explore each of these in turn.

#### 6.7.1 Double back slash starts on own line

We explore DBSStartsOnOwnLine ($\star$ in Listing 396); starting with the code in Listing 396, together with the YAML files given in Listing 398 and Listing 400 and running the following commands

```bash
$ latexindent -m tabular3.tex -l DBS1.yaml
$ latexindent -m tabular3.tex -l DBS2.yaml
```

then we receive the respective output given in Listing 397 and Listing 399.
We note that

- Listing 398 specifies `DBSStartsOnOwnLine` for every environment (that is within `lookForAlignDelims`, Section 5 on page 28); the double back slashes from Listing 396 have been moved to their own line in Listing 397;

- Listing 400 specifies `DBSStartsOnOwnLine` on a per-name basis for `tabular` (that is within `lookForAlignDelims`, Section 5 on page 28); the double back slashes from Listing 396 have been moved to their own line in Listing 399, having added comment symbols before moving them.

### 6.7.2 Double back slash finishes with line break

Let’s now explore `DBSFinishesWithLineBreak` (in Listing 396); starting with the code in Listing 396, together with the YAML files given in Listing 402 and Listing 404 and running the following commands:

```bash
cmh:~$ latexindent.pl -m tabular3.tex -l DBS3.yaml
```

then we receive the respective output given in Listing 401 and Listing 403.

We note that

- Listing 402 specifies `DBSFinishesWithLineBreak` for every environment (that is within `lookForAlignDelims`, Section 5 on page 28); the code following the double back slashes from Listing 396 has been moved to their own line in Listing 401;

- Listing 404 specifies `DBSFinishesWithLineBreak` on a per-name basis for `tabular` (that is within `lookForAlignDelims`, Section 5 on page 28); the first double back slashes from Listing 396 have moved code following them to their own line in Listing 403, 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.

### 6.7.3 Double back slash poly switches for specialBeginEnd

Let’s explore the double back slash poly-switches for code blocks within `specialBeginEnd` code blocks (Listing 83 on page 38); we begin with the code within Listing 405.
Upon using the YAML settings in Listing 407, and running the command

cmh:~$ latexindent.pl -m special4.tex -l DBS5.yaml

then we receive the output given in Listing 406.

There are a few things to note:

- in Listing 407 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.

6.7.4 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 408.

Upon using the YAML settings in Listings 410 and 412, 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 409 and 411.
### 6.7.5 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 413

```latex
\begin{pmatrix}
1 & 2 \\
\[2pt\] 3 & 4 \\
\[3 \text{ex}\] 5 & 6 \\
\[4 \text{pt}\] 7 & 8
\end{pmatrix}
```

and running the following command, using Listing 402,

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

then we receive the output given in Listing 414.

```latex
\begin{pmatrix}
1 & 2 \\
\[2pt\] 3 & 4 \\
\[3 \text{ex}\] 5 & 6 \\
\[4 \text{pt}\] 7 & 8
\end{pmatrix}
```

You can customise the pattern for the double back slash by exploring the `fine tuning` field detailed in Listing 470 on page 118.

### 6.8 Poly-switches for other code blocks

Rather than repeat the examples shown for the environment code blocks (in Section 6.6 on page 93), 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 51 on page 33) can not be modified using `latexindent.pl`. However, there are two poly-switches available for `verbatim` code blocks: environments (Listing 17 on page 26), commands (Listing 18 on page 26) and specialBeginEnd (Listing 96 on page 40).
### Table 2: Poly-switch mappings for all code-block types

<table>
<thead>
<tr>
<th>Code block</th>
<th>Sample</th>
<th>Poly-switch mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>environment</strong></td>
<td>before words♠, \begin{myenv}♥ heart, body of myenv♦, \end{myenv}♠ heart</td>
<td>♠ BeginStartsOnOwnLine, ♥ BodyStartsOnOwnLine, ♦ EndStartsOnOwnLine, ♣ EndFinishesWithLineBreak</td>
</tr>
</tbody>
</table>
| **ifelsefi**                  | before words♠, \if...♥ heart, body of if/or statement♦, \or♥ heart, body of if/or statement★, \else★, \fi★, \begin{verbatim}...
|                                | ✗ after words                                                          | ♠ IfStartsOnOwnLine, ♥ BodyStartsOnOwnLine, ♦ OrStartsOnOwnLine, ★ OrFinishesWithLineBreak, ★ ElseStartsOnOwnLine, ★ ElseFinishesWithLineBreak, ♦ FiStartsOnOwnLine, ♣ FiFinishesWithLineBreak |
| **optionalArguments**         | ...♠, [♥ heart, value before comma★, □, end of body of opt arg◇, ]♠ heart, ... |
|                                |                                                                      | ♠ LSqBStartsOnOwnLine, ♥ OptArgBodyStartsOnOwnLine, ★ CommaStartsOnOwnLine, ★ CommaFinishesWithLineBreak, ★ RsqBStartsOnOwnLine, ★ RsqBFinishesWithLineBreak |
| **mandatoryArguments**        | ...♠, {♥ heart, value before comma★, □, end of body of mand arg◇, }♠ heart, ... |
|                                |                                                                      | ♠ LCuBStartsOnOwnLine, ♥ MandArgBodyStartsOnOwnLine, ★ CommaStartsOnOwnLine, ★ CommaFinishesWithLineBreak, ★ RcubBStartsOnOwnLine, ★ RcubBFinishesWithLineBreak |
| **commands**                  | before words♠, \mycommand♥ heart, \begin{verbatim} ... |
|                                |                                                                      | ♠ CommandStartsOnOwnLine, ♥ CommandNameFinishesWithLineBreak |
| **namedGroupingBraces Brackets** | before words♠, myname♥ heart, (braces/brackets)                      | ♠ NameStartsOnOwnLine, ♥ NameFinishesWithLineBreak |
| **keyEqualsValuesBracesBrackets** | before words♠, key★ heart, =♥ heart, (braces/brackets)              | ♠ KeyStartsOnOwnLine, ♥ equalsStartsOnOwnLine, ♥ equalsFinishesWithLineBreak |
| **items**                     | before words♠, \item♥ heart, ...                                        | ♠ ItemStartsOnOwnLine, ♥ ItemFinishesWithLineBreak |
| **specialBeginEnd**           | before words♠, \begin{special}{middle}♥ heart, body of special/middle★, \middle★, body of special/middle ◇, \}★, after words | ♠ SpecialBeginStartsOnOwnLine, ♥ SpecialBodyStartsOnOwnLine, ★ SpecialMiddleStartsOnOwnLine, ★ SpecialMiddleFinishesWithLineBreak, ★ SpecialEndStartsOnOwnLine, ★ SpecialEndFinishesWithLineBreak |
| **verbatim**                  | before words♠, \begin{verbatim}...
|                                |                                                                      | ♠ VerbatimBeginStartsOnOwnLine |

---

9. LSqB stands for Left Square Bracket
10. LCuB stands for Left Curly Brace
6.9 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 previous page) and `LCuBStartsOnOwnLine` for mandatory arguments, and `LSqBStartsOnOwnLine` for optional arguments.

Let’s begin with the code in Listing 415 and the YAML settings in Listing 417; with reference to Table 2 on the preceding page, the key `CommandNameFinishesWithLineBreak` is an alias for `BodyStartsOnOwnLine`.

**LISTING 415: mycommand1.tex**
```
\mycommand
{ mand arg text
 mand arg text }
{ mand arg text
 mand arg text }
```

Upon running the command
```
cmh:~$ latexindent.pl -m -l=mycom-mlb1.yaml mycommand1.tex
```

we obtain Listing 416; note that the second mandatory argument beginning brace `{` has had its leading line break removed, but that the first brace has not.

**LISTING 416: mycommand1.tex using Listing 417**
```
\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 419; upon running the analogous command to that given above, we obtain Listing 418; both beginning braces `{` have had their leading line breaks removed.

**LISTING 418: mycommand1.tex using Listing 419**
```
\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 421; upon running the analogous command to that given above, we obtain Listing 420.
6.10 Conflicting poly-switches: sequential code blocks

It is very easy to have conflicting poly-switches; if we use the example from Listing 415 on the previous page, and consider the YAML settings given in Listing 423. The output from running:

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

is given in Listing 423.

Studying Listing 423, 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 422, 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 425; upon running the command:

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

we obtain the output given in Listing 424.

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 427, which give associated output in Listing 426.
6.11 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 428, noting that it contains nested environments.

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

Let's use the YAML settings given in Listing 430, which upon running the command

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

gives the output in Listing 429.

In Listing 429, let's first of all note that both environments have received the appropriate (default) indentation; secondly, note that the poly-switch EndStartsOnOwnLine appears to have won the conflict, as \end{one} has had its leading line break removed.

To understand it, let's talk about the three basic phases of \texttt{latexindent.pl}:

1. Phase 1: packing, in which code blocks are replaced with unique ids, working from \textit{the inside to the outside}, and then sequentially – for example, in Listing 428, 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 \texttt{EndFinishesWithLineBreak} is 1 or 2).

2. Phase 2: indentation, in which white space is added to the begin, body, and end statements;

3. Phase 3: unpacking, in which unique ids are replaced by their \textit{indented} code blocks; if the \texttt{-m} switch is active, then during this phase,
   • line breaks before \texttt{begin} statements can be added or removed (depending upon \texttt{BeginStartsOnOwnLine});
   • line breaks after \texttt{end} statements can be removed but NOT added (see \texttt{EndFinishesWithLineBreak}).

With reference to Listing 429, this means that during Phase 1:

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

• next, the \texttt{one} environment is found; the line break ahead of \texttt{\end{one}} is removed because \texttt{EndStartsOnOwnLine} is set to \texttt{-1}.

The indentation is done in Phase 2; in Phase 3 \textit{there is no option to add a line break after the end statements}. We can justify this by remembering that during Phase 3, the \texttt{one} environment will be found and processed first, followed by the \texttt{two} environment. If the \texttt{two} environment were to add a line break after the \texttt{\end{two}} statement, then \texttt{latexindent.pl} would have no way of knowing how much indentation to add to the subsequent text (in this case, \texttt{\end{one}}).

We can explore this further using the poly-switches in Listing 432; upon running the command

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

we obtain the output given in Listing 431.

**Listing 431:** nested-env.tex using Listing 432

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

During Phase 1:

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

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

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


**SECTION 7**

## The -r, -rv and -rr switches

You can instruct `latexindent.pl` to perform replacements/substitutions on your file by using any of the -r, -rv or -rr switches:

- the -r switch will perform indentation and replacements, not respecting verbatim code blocks;
- the -rv switch will perform indentation and replacements, and will respect verbatim code blocks;
- the -rr switch will *not* perform indentation, and will perform replacements not respecting verbatim code blocks.

We will demonstrate each of the -r, -rv and -rr switches, but a summary is given in Table 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 replacements field is shown in Listing 433; as with all of the other fields, you are encouraged to customise and charge 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 433.

```yaml
replacements:
- amalgamate: 1
```

The first entry within the replacements field is *amalgamate*, and is optional; by default it is set to 1, so that replacements will be amalgamated from each settings file that you specify. As you’ll see in the demonstrations that follow, there is no need to specify this field.

You’ll notice that, by default, there is only *one* entry in the replacements field, but it can take as many entries as you would like; each one needs to begin with a - on its own line.

### 7.1 Introduction to replacements

Let’s explore the action of the default settings, and then we’ll demonstrate the feature with further examples. With reference to Listing 433, the default action will replace every instance of the text `latexindent.pl` with `pl.latexindent`.

Beginning with the code in Listing 434 and running the command

```
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 438

```
\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 440, and running the command

```
cmh:~$ latexindent.pl -r colsep.tex -l=colsep.yaml
```
then we achieve the output in Listing 439.

Listing 439: colsep.tex using Listing 438

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

Listing 440: colsep.yaml

replacements:
  -
    this: \arraycolsep=3pt
  -
    this: \arraycolsep=5pt

Note that in Listing 440, 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 440 more concise by exploring the substitution field. Using the settings in Listing 442 and running the command

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

then we achieve the output in Listing 441.

Listing 441: colsep.tex using Listing 442

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

Listing 442: colsep1.yaml

replacements:
  -
    substitution: s/\arraycolsep=\d+pt//sg

The code given in Listing 442 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, [7] for a detailed covering of the topic. With reference to Listing 442, 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 \textit{sg} at the end of the line) instructs latexindent.pl to treat your file as one single line;

• the $g$ modifier (in the \textit{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 442 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 \textit{sg} modifiers for this feature.

Example 2 We’ll keep working with the file in Listing 438 on the previous page for this example.

Using the YAML in Listing 444, and running the command

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

then we achieve the output in Listing 443.
With reference to Listing 444, we have specified a multi-line version of this by employing the literal YAML style !-. 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 when field, specified in Listing 433 on page 109. This field can take two values: before and after, which respectively instruct latexindent.pl to perform the replacements before indentation or after it. The default value is before.

Using the YAML in Listing 446, and running the command

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

then we achieve the output in Listing 445.

We note that, because we have specified when: after, that latexindent.pl has not found the string specified in Listing 446 within the file in Listing 438 on page 110. As it has looked for the string within Listing 446 after the indentation has been performed. After indentation, the string as written in Listing 446 is no longer part of the file, and has therefore not been replaced.

As a final note on this example, if you use the -rr switch, as follows,

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

then the when field is ignored, no indentation is done, and the output is as in Listing 443.

**Example 3** An important part of the substitution routine is in *capture groups*.

Assuming that we start with the code in Listing 447, let’s assume that our goal is to replace each occurrence of $$...$$ with \begin{equation*}...\end{equation*}. This example is partly motivated by tex stackexchange question 242150.
7.3 Examples of replacements

LISTING 447: 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 449 and run the command

```
cmh:~$ latexindent.pl -r displaymath.tex -l=displaymath1.yaml
```
to receive the output given in Listing 448.

LISTING 448: displaymath.tex using Listing 449

before text \begin{equation*}a^2+b^2=4\end{equation*} and
\begin{equation*}c^2\end{equation*}
\begin{equation*}d^2+e^2 = f^2\end{equation*}
and also \begin{equation*}g^2\end{equation*} and some inline math: $h^2$

A few notes about Listing 449:

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 6.5 on page 92, which we do in Listing 451; upon running the command

```
cmh:~$ latexindent.pl -r -m displaymath.tex -l=displaymath1.yaml, equation.yaml
```
then we receive the output in Listing 450.
7.3 Examples of replacements

**Example 4** This example is motivated by tex stackexchange question 490086. We begin with the code in Listing 452.

```latex
\begin{equation*}
\begin{align*}
 a^2 + b^2 &= 4 \\
 c^2 \\
 d^2 + e^2 &= f^2 \\
 g^2
\end{align*}
\end{equation*}
and some inline math: $h^2$
```

Our goal is to make the spacing uniform between the phrases. To achieve this, we employ the settings in Listing 454, and run the command

```
cmh: ∼ $latexindent.pl -r phrase.tex -l=hspace.yaml
```

which gives the output in Listing 453.

```latex
\begin{center}
\begin{tabular}{ccc}
\textbf{phrase 1} & \textbf{phrase 2} & \textbf{phrase 3} & \textbf{phrase 100} \\
\textbf{phrase 1} & \textbf{phrase 2} & \textbf{phrase 3} & \textbf{phrase 100} \\
\textbf{phrase 1} & \textbf{phrase 2} & \textbf{phrase 3} & \textbf{phrase 100} \\
\textbf{phrase 1} & \textbf{phrase 2} & \textbf{phrase 3} & \textbf{phrase 100} \\
\textbf{phrase 1} & \textbf{phrase 2} & \textbf{phrase 3} & \textbf{phrase 100} \\
\end{tabular}
\end{center}
```

The \h+ setting in Listing 454 say to replace at least one horizontal space with a single space.
Example 5  We begin with the code in Listing 455.

```
<table>
<thead>
<tr>
<th>LISTING 455: references.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{equation \eqref{eq:aa} and Figure \ref{fig:bb}}</td>
</tr>
<tr>
<td>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 457 and running the command

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

which gives the output in Listing 456.

```
<table>
<thead>
<tr>
<th>LISTING 456: references.tex using Listing 457</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{\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 457: reference.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>replacements:</td>
</tr>
</tbody>
</table>
| - substitution: |-
| s/( |
|.equation |
| |table |
| |figure |
| |section |
| ) |
|\h|-)* |
|\(?:eq)? |
|ref\{(.*?)\}/\hyperref{$1 \ref\*{$3}}/sgxi |
```

Referencing Listing 457, 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 109) in the context of an example that contains a verbatim code block, Listing 458; we will use the settings in Listing 459.

```
<table>
<thead>
<tr>
<th>LISTING 458: verb1.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{\begin{myenv}body of verbatim\end{myenv}}</td>
</tr>
<tr>
<td>some verbatim</td>
</tr>
<tr>
<td>\texttt{\begin{verbatim}body of verbatim text\end{verbatim}}</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>LISTING 459: 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 receive the respective output in Listings 460 to 462.

#### Listing 460: verb1-mod1.tex
```
\begin{myenv}
  head of verbatim
\end{myenv}

\begin{verbatim}
  head of verbatim
  of verbatim
  text
\end{verbatim}

text
```

#### Listing 461: verb1-rv-mod1.tex
```
\begin{myenv}
  head of verbatim
\end{myenv}

\begin{verbatim}
  body of verbatim
  text
\end{verbatim}

\begin{verbatim}
  head of verbatim
  text
\end{verbatim}
```

#### Listing 462: verb1-rr-mod1.tex
```
\begin{myenv}
  head of verbatim
\end{myenv}

\begin{verbatim}
  head of verbatim
  text
\end{verbatim}

\begin{verbatim}
  text
\end{verbatim}
```

We note that:

1. in Listing 460 indentation has been performed, and that the replacements specified in Listing 459 have been performed, even within the verbatim code block;
2. in Listing 461 indentation has been performed, but that the replacements have not been performed within the verbatim environment, because the \texttt{rv} switch is active;
3. in Listing 462 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 109.

#### Example 7

Let's explore the \texttt{amalgamate} field from Listing 433 on page 109 in the context of the file specified in Listing 463.

#### Listing 463: amalg1.tex
```
one two three
```

Let's consider the YAML files given in Listings 464 to 466.

#### Listing 464: amalg1-yaml.yaml
```
- this: one
  that: 1
```

#### Listing 465: amalg2-yaml.yaml
```
- this: two
  that: 2
```

#### Listing 466: amalg3-yaml.yaml
```
- amalgamate: 0
- this: three
  that: 3
```

Upon running the following commands,
```
cmh:~$ latexindent.pl -r amalg1.tex -l=amalg1-yaml
```
```
cmh:~$ latexindent.pl -r amalg1.tex -l=amalg1-yaml,amalg2-yaml
```
```
cmh:~$ latexindent.pl -r amalg1.tex -l=amalg1-yaml,amalg2-yaml,amalg3-yaml
```

we receive the respective output in Listings 467 to 469.
### 7.3 Examples of replacements

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1 two three</td>
<td>1 2 three</td>
<td>one two 3</td>
</tr>
</tbody>
</table>

We note that:

1. in Listing 467 the replacements from Listing 464 have been used;
2. in Listing 468 the replacements from Listings 464 and 465 have both been used, because the default value of `amalgamate` is 1;
3. in Listing 469 only the replacements from Listing 466 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 44. The fine tuning of the details of such code blocks is controlled by the `fineTuning` field, detailed in Listing 470. 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 470 are all regular expressions. This manual is not intended to be a tutorial on regular expressions; you might like to read, for example, [7] for a detailed covering of the topic.

We make the following comments with reference to Listing 470:

1. the `environments:name` field details that the name of an environment can contain:
   1a. a-z lower case letters
   1b. A-Z upper case letters
   1c. @ the @ 'letter'
   1d. * stars
   1e. 0-9 numbers
   1f. _ underscores
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 66. In particular:
   (a) betterFullStop is in relation to the one sentence per line routine, detailed in Section 6.2 on page 76
   (b) doubleBackSlash is in relation to the DBSStartsOnOwnLine and DBSFinishesWithLineBreak polyswitches surrounding double back slashes, see Section 6.7 on page 100
   (c) comma is in relation to the CommaStartsOnOwnLine and CommaFinishesWithLineBreak polyswitches surrounding commas in optional and mandatory arguments; see Table 2 on page 104

It is not obvious from Listing 470, 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 471, together with its default output using the command

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

is given in Listing 472.

<table>
<thead>
<tr>
<th>LISTING 471: finetuning1.tex</th>
<th>LISTING 472: finetuning1.tex default</th>
</tr>
</thead>
<tbody>
<tr>
<td>\mycommand{</td>
<td>\mycommand{</td>
</tr>
<tr>
<td>\rule{G -&gt; +H[-G]CL}</td>
<td>\rule{G -&gt; +H[-G]CL}</td>
</tr>
<tr>
<td>\rule{H -&gt; -G+[H]CL}</td>
<td>\rule{H -&gt; -G+[H]CL}</td>
</tr>
<tr>
<td>\rule{g -&gt; +h[-g]cL}</td>
<td>\rule{g -&gt; +h[-g]cL}</td>
</tr>
<tr>
<td>\rule{h -&gt; -g[+h]cL}</td>
<td>\rule{h -&gt; -g[+h]cL}</td>
</tr>
<tr>
<td>}</td>
<td>}</td>
</tr>
</tbody>
</table>

It's clear from Listing 472 that the indentation scheme has not worked as expected. We can fine tune the indentation scheme by employing the settings given in Listing 474 and running the
Example 9

Let’s have another demonstration; consider the file given in Listing 475, together with its default output using the command

```
cmh:~$ latexindent.pl finetuning2.tex
```

is given in Listing 476.

It’s clear from Listing 476 that the indentation scheme has not worked as expected. We can fine tune the indentation scheme by employing the settings given in Listing 478 and running the command

```
cmh:~$ latexindent.pl finetuning2.tex -l=fine-tuning2.yaml
```

and the associated (desired) output is given in Listing 477.

In particular, note that the settings in Listing 478 specify that NamedGroupingBracesBrackets and UnNamedGroupingBracesBrackets can follow " and that we allow \- between arguments.
SECTION 9

Conclusions and known limitations

There are a number of known limitations of the script, and almost certainly quite a few that are unknown!

The main limitation is to do with the alignment routine discussed on page 28; for example, consider the file given in Listing 479.

\begin{verbatim}
\matrix (A) {
  c01 & c02 & c03 & c0q \\
  c_{11} & c12 & \ldots & c1q \\
};
\end{verbatim}

The default output is given in Listing 480, and it is clear that the alignment routine has not worked as hoped, but it is expected.

\begin{verbatim}
\matrix (A) {
  c01 & c02 & c03 & c0q \\
  c_{11} & c12 & \ldots & c1q \\
};
\end{verbatim}

The reason for the problem is that when \texttt{latexindent.pl} stores its code blocks (see Table 1 on page 44) it uses replacement tokens. The alignment routine is using the length of the replacement token in its measuring – I hope to be able to address this in the future.

There are other limitations to do with the multicolumn alignment routine (see Listing 39 on page 30); in particular, when working with code blocks in which multicolumn commands overlap, the algorithm can fail.

Another limitation is to do with efficiency, particularly when the -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 107); it is hoped 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 15 on page 24) will be consulted. If you find a case in which the script struggles, please feel free to report it at \cite{8}, and in the meantime, consider using a \texttt{noIndentBlock} (see page 26).

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 \cite{8}; otherwise, feel free to find me on the \url{http://tex.stackexchange.com/users/6621/cmhughes}. 
References

10.1 External links


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 481 (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.

```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 Getopt::Long;
use Data::Dumper;
use List::Util qw(max);
use Log::Log4perl qw(get_logger :levels);

print "hello\nworld";
exit;
```

A.1 Module installer script

latexindent.pl ships with a helper script that will install any missing perl modules on your system; if you run

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

or

```
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 481 will vary depending on your operating system and Perl distribution. For example, Ubuntu users might visit the software center, or else run
A.2 Manually installed modules

They may be interested in exploring Perl[14]; possible installation and setup options follow for Ubuntu (other distributions will need slightly different commands).

```bash
cmh:~$ sudo perl -MCPAN -e 'install,"File::HomeDir"'
```

Linux users may be interested in exploring Perl[14]; possible installation and setup options follow for Ubuntu (other distributions will need slightly different commands).

```bash
cmh:~$ sudo apt-get install perlbrew
cmh:~$ perlbrew init
```  
```bash
cmh:~$ perlbrew install perl-5.28.1
```  
```bash
cmh:~$ perlbrew switch perl-5.28.1
```  
```bash
cmh:~$ sudo apt-get install curl
```  
```bash
cmh:~$ curl -L http://cpanmin.us | perl - App::cpanminus
```  
```bash
cmh:~$ cpanm YAML::Tiny
```  
```bash
cmh:~$ cpanm File::HomeDir
```  
```bash
cmh:~$ cpanm Unicode::GCString
```  
```bash
cmh:~$ cpanm Log::Log4perl
```  
```bash
cmh:~$ cpanm Log::Dispatch
```

Users of the Macintosh operating system might like to explore the following commands, for example:

```bash
cmh:~$ brew install perl
```  
```bash
cmh:~$ brew install cpanm
```  
```bash
cmh:~$ cpanm YAML::Tiny
```  
```bash
cmh:~$ cpanm File::HomeDir
```  
```bash
cmh:~$ cpanm Unicode::GCString
```  
```bash
cmh:~$ cpanm Log::Log4perl
```  
```bash
cmh:~$ cpanm Log::Dispatch
```

Strawberry Perl users on Windows might use CPAN client. All of the modules are readily available on CPAN [4].

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
```
SECTION B

Updating the path variable

\texttt{latexindent.pl} has a few scripts (available at [8]) that can update the path variables. Thank you to [9] for this feature. If you’re on a Linux or Mac machine, then you’ll want \texttt{CMakeLists.txt} from [8].

B.1 Add to path for Linux

To add \texttt{latexindent.pl} to the path for Linux, follow these steps:

1. download \texttt{latexindent.pl} and its associated modules, \texttt{defaultSettings.yaml}, to your chosen directory from [8];

2. within your directory, create a directory called \texttt{path-helper-files} and download \texttt{CMakeLists.txt} and \texttt{cmake_uninstall.cmake.in} from [8]/\texttt{path-helper-files} to this directory;

3. run

\begin{verbatim}
cmh:~$ ls /usr/local/bin
\end{verbatim}

to see what is \textit{currently} in there;

4. run the following commands

\begin{verbatim}
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
\end{verbatim}

5. run

\begin{verbatim}
cmh:~$ ls /usr/local/bin
\end{verbatim}

again to check that \texttt{latexindent.pl}, its modules and \texttt{defaultSettings.yaml} have been added.

To remove the files, run

\begin{verbatim}
cmh:~$ sudo make uninstall
\end{verbatim}

B.2 Add to path for Windows

To add \texttt{latexindent.exe} to the path for Windows, follow these steps:

1. download \texttt{latexindent.exe}, \texttt{defaultSettings.yaml}, \texttt{add-to-path.bat} from [8] to your chosen directory;

2. open a command prompt and run the following command to see what is \textit{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 16 on page 25 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 482, and the settings specified in Listing 483.

<table>
<thead>
<tr>
<th>Listing 482: simple.tex</th>
<th>Listing 483: logfile-prefs1.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{myenv}</td>
<td>logFilePreferences:</td>
</tr>
<tr>
<td>body of myenv</td>
<td>\begin{yaml}</td>
</tr>
<tr>
<td>\end{myenv}</td>
<td>\item showDecorationStartCodeBlockTrace: &quot;+++++&quot;</td>
</tr>
<tr>
<td></td>
<td>\item showDecorationFinishCodeBlockTrace: &quot;-----&quot;</td>
</tr>
</tbody>
</table>

If we run the following command (noting that \texttt{-t} is active)

```bash
$ latexindent.pl -t -l=logfile-prefs1.yaml simple.tex
```

then on inspection of indent.log we will find the snippet given in Listing 484.

<table>
<thead>
<tr>
<th>Listing 484: indent.log</th>
</tr>
</thead>
<tbody>
<tr>
<td>++++</td>
</tr>
<tr>
<td>TRACE: environment found: myenv</td>
</tr>
<tr>
<td>No ancestors found for myenv</td>
</tr>
<tr>
<td>Storing settings for myenv</td>
</tr>
<tr>
<td>indentRulesGlobal specified (0) for environments, ...</td>
</tr>
<tr>
<td>Using defaultIndent for myenv</td>
</tr>
<tr>
<td>Putting linebreak after replacementText for myenv</td>
</tr>
<tr>
<td>looking for COMMANDS and key = {value}</td>
</tr>
<tr>
<td>TRACE: Searching for commands with optional and/or mandatory arguments AND key = {value}</td>
</tr>
<tr>
<td>looking for SPECIAL begin/end</td>
</tr>
<tr>
<td>TRACE: Searching myenv for special begin/end (see specialBeginEnd)</td>
</tr>
<tr>
<td>TRACE: Searching myenv for optional and mandatory arguments</td>
</tr>
<tr>
<td>... no arguments found</td>
</tr>
<tr>
<td>-----</td>
</tr>
</tbody>
</table>

Notice that the information given about \texttt{myenv} is ‘framed’ using ++++ and ----- respectively.
SECTION D

Differences from Version 2.2 to 3.0

There are a few (small) changes to the interface when comparing Version 2.2 to Version 3.0. Explicitly, in previous versions you might have run, for example,

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

whereas in Version 3.0 you would run any of the following, for example,

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

cmh:~$ latexindent.pl -o outputfile.tex myfile.tex

cmh:~$ latexindent.pl myfile.tex -o outputfile.tex

cmh:~$ latexindent.pl myfile.tex -outputfile=outputfile.tex

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

noting that the output file is given next to the -o switch.

The fields given in Listing 485 are obsolete from Version 3.0 onwards.

```
alwaysLookforSplitBrackets
alwaysLookforSplitBrackets
checkunmatched
checkunmatchedELSE
checkunmatchedbracket
constructIfElseFi
```

There is a slight difference when specifying indentation after headings; specifically, we now write `indentAfterThisHeading` instead of `indent`. See Listings 486 and 487

```
LISTING 486: indentAfterThisHeading in Version 2.2
indentAfterHeadings:
  part:
    indent: 0
    level: 1

LISTING 487: indentAfterThisHeading in Version 3.0
indentAfterHeadings:
  part:
    indentAfterThisHeading: 0
    level: 1
```

To specify `noAdditionalIndent` for display-math environments in Version 2.2, you would write YAML as in Listing 488; as of Version 3.0, you would write YAML as in Listing 489 or, if you're using -m switch, Listing 490.
LISTING 488: noAdditionalIndent in Version 2.2

noAdditionalIndent:
\[: 0
\]: 0

LISTING 489: noAdditionalIndent for displayMath in Version 3.0

specialBeginEnd:
    displayMath:
        begin: '\\['
        end: '\\']'
    lookForThis: 0

LISTING 490: noAdditionalIndent for displayMath in Version 3.0

noAdditionalIndent:
    displayMath: 1

End