latexindent.pl

Version 2.2

Chris Hughes *

May 28, 2016

Abstract

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 indentation after headings (such as \chapter, \section, etc) are also available.

Contents

*and contributors! (See **??** on page ??.)

Listings

1 Before we begin

1.1 Thanks

I first created `latexindent.pl` to help me format chapter files in a big project. After I blogged about it on the TEX stack exchange [**cmhblog**] I received some positive feedback and follow-up feature requests. A big thank you to Harish Kumar who has really helped to drive the script forward and has put it through a number of challenging tests–I look forward to more challenges in the future Harish!

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

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 **??** on page ?? for their valued contributions.

1.2 License

`latexindent.pl` is free and open source, and it always will be. Before you start using it on any important files, bear in mind that `latexindent.pl` has the option to overwrite your `.tex` files. It will always make at least one backup (you can choose how many it makes, see page 7) 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 5). You, the user, are responsible for ensuring that you maintain backups of your files before running `latexindent.pl` on them. I think it is important at this stage to restate an important part of the license here:

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

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

Before you try the script on anything important (like your thesis), test it out on the sample files that come with it in the `success` directory.

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 [**cmh:videodemo**]

As you look at Listings 1 to 6, remember that `latexindent.pl` is just following its rules–there is nothing particular about these code snippets. All of the rules can be modified so that each user can personalize their indentation scheme.

In each of the samples given in Listings 1 to 6 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 4).

LISTING 1: `filecontents` before

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

LISTING 2: `filecontents` after

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

LISTING 3: `tikzset` before

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

LISTING 4: `tikzset` after

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

LISTING 5: `pstricks` before

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

LISTING 6: `pstricks` after

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

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 MiKTEXdistributions for Windows users. These files are also available from github [**latexindent-home**] should you wish to use them without a TEX distribution; in this case, you may like to read appendix C on page 21 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 4 on page 7.

`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 19 for details.

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. 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.

`latexindent.pl`

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

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

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

`latexindent.pl myfile.tex`

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

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

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 4; 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.

`tex,--outputfile=output.tex` `latexindent.pl -o=output.tex myfile.tex`

This will indent `myfile.tex` and output it to `output.tex`, overwriting it (`output.tex`) if it already exists. Note that if `latexindent.pl` is called with both the `-w` and `-o` switches, then `-w` will be ignored and `-o` will take priority (this seems safer than the other way round).

Note that using `-o` is equivalent to using `latexindent.pl myfile.tex > output.tex`

`-s, --silent` `latexindent.pl -s myfile.tex`

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

`-t, --trace` `latexindent.pl -t myfile.tex`

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.

`-tt, --ttrace` `latexindent.pl -tt myfile.tex`

More detailed tracing mode: this option gives more details to `indent.log` than the standard `trace` option.

`-l, --local[=myyaml.yaml]` `latexindent.pl -l myfile.tex`

`latexindent.pl -l=myyaml.yaml myfile.tex`

`latexindent.pl -l myyaml.yaml myfile.tex`

Local settings: you might like to read Section 4 before using this switch. `latexindent.pl` will always load `defaultSettings.yaml` and if it is called with the `-l` 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 of a `yaml` file 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`.

`-d, --onlydefault` `latexindent.pl -d myfile.tex`

Only `defaultSettings.yaml`: you might like to read Section 4 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`/`.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.

`-c, --cruft=<directory>` `latexindent.pl -c=/path/to/directory/ myfile.tex`

If you wish to have backup files and `indent.log` written to a directory other than the current working directory, then you can send these 'cruft' files to another directory.

`latexindent.pl` can also be called on a file without the file extension, for example `latexindent.pl myfile` and in which case, you can specify the order in which extensions are searched for; see Listing 29 on page 13 for full details.

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`. `latexindent.pl` ships with an `arara` rule, `indent.yaml`, which can be copied to the directory of your other `arara` rules; otherwise you can add the directory in which `latexindent.pl` resides to your `araraconfig.yaml` file.

Once you have told `arara` where to find your `indent` rule, you can use it any of the ways described in Listing 7 (or combinations thereof). In fact, `arara` allows yet greater flexibility– you can use `yes/no`, `true/false`, or `on/off` to toggle the various options.

LISTING 7: `arara` sample usage

```
% arara: indent
% arara: indent: {overwrite: yes}
% arara: indent: {output: myfile.tex}
% arara: indent: {silent: yes}
% arara: indent: {trace: yes}
% arara: indent: {localSettings: yes}
% arara: indent: {onlyDefault: on}
% arara: indent: { cruft: /home/cmhughes/Desktop }
\documentclass{article}
...
```

Hopefully the use of these rules is fairly self-explanatory, but for completeness Table 1 shows the relationship between `arara` directive arguments and the switches given in Section 3.1.

TABLE 1: `arara` directive arguments and corresponding switches

`arara` directive argument	switch
`overwrite`	`-w`
`output`	`-o`
`silent`	`-s`
`trace`	`-t`
`localSettings`	`-l`
`onlyDefault`	`-d`
`cruft`	`-c`

The `cruft` directive does not work well when used with directories that contain spaces.

4 default, user, and local settings

`latexindent.pl` loads its settings from `defaultSettings.yaml` (rhymes with camel). 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.

4.1 `defaultSettings.yaml`

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.

You can certainly feel free to edit `defaultSettings.yaml`, but this is not ideal as it may be overwritten when you update your TeX distribution – all of your hard work tweaking the script would be undone! Don't worry, there's a solution, feel free to peek ahead to Section 4.2 if you like.

`defaultIndent "\t"`

This is the default indentation (`\t` means a tab) used in the absence of other details for the command or environment we are working with–see `indentRules` for more details (page 11).

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

`backupExtension .bak`

If you call `latexindent.pl` with the `-w` switch (to overwrite `myfile.tex`) then it will create a backup file before doing any indentation: `myfile.bak0`

By default, every time you call `latexindent.pl` after this with the `-w` switch it will create `myfile.bak1`, `myfile.bak2`, etc.

`onlyOneBackUp 0`

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`.

`maxNumberOfBackUps` 0

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`.

`cycleThroughBackUps` 0

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 in Listing 8 would be obeyed.

LISTING 8: `cycleThroughBackUps`

```
copy myfile.bak1 to myfile.bak0
copy myfile.bak2 to myfile.bak1
copy myfile.bak3 to myfile.bak2
copy myfile.bak4 to myfile.bak3
```

`indentPreamble` 0

The preamble of a document can sometimes contain some trickier code for `latexindent.pl` to work with. By default, `latexindent.pl` won't try to operate on the preamble, but if you'd like it to try then change `indentPreamble` to `1`.

`alwaysLookforSplitBraces` 1

This switch tells `latexindent.pl` to look for commands that can split *braces* across lines, such as `parbox`, `tikzset`, etc. In older versions of `latexindent.pl` you had to specify each one in `checkunmatched`–this clearly became tedious, hence the introduction of `alwaysLookforSplitBraces`.

As long as you leave this switch on (set to 1) you don't need to specify which commands can split braces across lines–you can ignore the fields `checkunmatched` *and* `checkunmatchedELSE` *described later on page 14.*

`alwaysLookforSplitBrackets` 1

This switch tells `latexindent.pl` to look for commands that can split *brackets* across lines, such as `psSolid`, `pgfplotstabletypeset`, etc. In older versions of `latexindent.pl` you had to specify each one in `checkunmatchedbracket`– this clearly became tedious, hence the introduction of `alwaysLookforSplitBraces`.

As long as you leave this switch on (set to 1) you don't need to specify which commands can split brackets across lines–you can ignore `checkunmatchedbracket` *described later on page 14.*

`removeTrailingWhitespace` 0

By default `latexindent.pl` indents every line (including empty lines) which creates 'trailing white space' feared by most version control systems. If this option is set to `1`, trailing white space is removed from all lines, also non-empty ones. In general this should not create any problems, but by precaution this option is turned off by default. Thanks to [**vosskuhle**] for providing this feature.

`lookForAlignDelims` This is the first example of a field in `defaultSettings.yaml` that has more than one line; Listing 9 shows more details. In fact, the fields in `lookForAlignDelims` can actually take two different forms: the *basic* version is shown in Listing 9 and the *advanced* version in Listing 12; we will discuss each in turn.

LISTING 9: `lookForAlignDelims` (basic)

```
lookForAlignDelims:
   tabular: 1
   tabularx: 1
   longtable: 1
   array: 1
   matrix: 1
   bmatrix: 1
   pmatrix: 1
   align: 1
   align*: 1
   alignat: 1
   alignat*: 1
   aligned: 1
   cases: 1
   dcases: 1
   pmatrix: 1
   listabla: 1
```

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 5), but in many cases it will produce results such as those in Listings 10 and 11.

LISTING 10: `tabular` before

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

LISTING 11: `tabular` after (basic)

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

If you find that `latexindent.pl` does not perform satisfactorily on such environments then you can either remove them from `lookForAlignDelims` altogether, or set the relevant key to `0`, for example `tabular: 0`, or if you just want to ignore *specific* instances of the environment, you could wrap them in something from `noIndentBlock` (see Listing 19).

If you wish to remove the alignment of the `\\` within a delimiter-aligned block, then the advanced form of `lookForAlignDelims` shown in Listing 12 is for you.

LISTING 12: `lookForAlignDelims` (advanced)

```
lookForAlignDelims:
   tabular:
      delims: 1
      alignDoubleBackSlash: 0
      spacesBeforeDoubleBackSlash: 0
   tabularx:
      delims: 1
   longtable: 1
```

Note that you can use a mixture of the basic and advanced form: in Listing 12 `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 up to 3 fields:

- `delims`: switch equivalent to simply specifying, for example, `tabular: 1` in the basic version shown in Listing 9 (default: 1);
- `alignDoubleBackSlash`: switch to determine if \\ should be aligned (default: 1);
- `spacesBeforeDoubleBackSlash`: optionally, specifies the number of spaces to be inserted before (non-aligned) \\. In order to use this field, `alignDoubleBackSlash` needs to be set to 0 (default: 0).

With the settings shown in Listing 12 we receive the before-and-after results shown in Listings 13 and 14; note that the ampersands have been aligned, but the \\ have not (compare the alignment of \\ in Listings 11 and 14).

LISTING 13: `tabular` before

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

LISTING 14: `tabular` after (advanced)

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

Using `spacesBeforeDoubleBackSlash: 3` gives Listings 15 and 16, note the spacing before the \\ in Listing 16.

LISTING 15: `tabular` before

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

LISTING 16: `tabular` after (spacing)

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

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 use the mark up illustrated in Listing 17. 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`.

LISTING 17: Mark up for aligning delimiters outside of environments

```
\matrix{%
%* \begin{tabular}
 1 & 2 & 3 & 4 \\
 5 &   & 6 &   \\
%* \end{tabular}
}
```

`verbatimEnvironments` A field that contains a list of environments that you would like left completely alone–no indentation will be done to environments that you have specified in this field–see Listing 18.

LISTING 18: `verbatimEnvironments`

```
1  verbatimEnvironments:
2      verbatim: 1
3      lstlisting: 1
```

Note that if you put an environment in `verbatimEnvironments` and in other fields such

as `lookForAlignDelims` or `noAdditionalIndent` then `latexindent.pl` will *always* prioritize `verbatimEnvironments`.

`noIndentBlock` 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.

LISTING 19: `noIndentBlock`

```
noIndentBlock:
    noindent: 1
    cmhtest: 1
```

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.

LISTING 20: `noIndentBlock` demonstration

```
% \begin{noindent}
       this code
              won't
    be touched
                 by
           latexindent.pl!
%\end{noindent}
```

`noAdditionalIndent` If you would prefer some of your environments or commands not to receive any additional indent, then populate `noAdditionalIndent`; see Listing 21. Note that these environments will still receive the *current* level of indentation unless they belong to `verbatimEnvironments`, or `noIndentBlock`.

LISTING 21: `noAdditionalIndent`

```
noAdditionalIndent:
    document: 1
    myexample: 1
    mydefinition: 1
    problem: 1
    exercises: 1
    mysolution: 1
    foreach: 0
    widepage: 1
    comment: 1
    \[: 1
    \]: 1
    frame: 0
```

Note in particular from Listing 21 that if you wish content within `\[` and `\]` to receive no additional indentation then you have to specify *both* as `1` (the default is `0`). If you do not specify both as the same value you may get some interesting results!

`indentRules` If you would prefer to specify individual rules for certain environments or commands, just populate `indentRules`; see Listing 22

LISTING 22: indentRules

```
1  indentRules:
2     myenvironment: "\t\t"
3     anotherenvironment: "\t\t\t\t"
4     \[: "\t"
```

Note that in contrast to `noAdditionalIndent` you do *not* need to specify both `\[` and `\]` in this field.

If you put an environment in both `noAdditionalIndent` and in `indentRules` then `latexindent.pl` will resolve the conflict by ignoring `indentRules` and prioritizing `noAdditionalIndent`. You will get a warning message in `indent.log`; note that you will only get one warning message per command or environment. Further discussion is given in Section 4.1.1.

`indentAfterHeadings` This field enables the user to specify indentation rules that take effect after heading commands such as `\part`, `\chapter`, `\section`, `\subsection*` etc. This field is slightly different from most of the fields that we have considered previously, because each element is itself a field which has two elements: `indent` and `level`. (Similar in structure to the advanced form of `lookForAlignDelims` in Listing 12.)

LISTING 23: indentAfterHeadings

```
1  indentAfterHeadings:
2      part:
3         indent: 0
4         level: 1
5      chapter:
6         indent: 0
7         level: 2
8      section:
9         indent: 0
10        level: 3
11     ...
```

The default settings do *not* place indentation after a heading–you can easily switch them on by changing `indent: 0` to `indent: 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`– you will find the default `indentRules` contains `chapter: " "` which tells `latexindent.pl` simply to use a space character after `\chapter` headings (once `indent` is set to 1 for `chapter`).

`indentAfterItems` The environments 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`.

LISTING 24: indentAfterItems

```
indentAfterItems:
    itemize: 1
    enumerate: 1
```

A demonstration is given in Listings 25 and 26

LISTING 25: items before

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

LISTING 26: items after

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

`itemNames` 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 4.2 on page 16 for details of how to configure user settings, and Listing 38 on page 17 in particular.

`constructIfElseFi` The commands specified in this field will tell `latexindent.pl` to look for constructs that have the form `\if... \else... \fi`, such as, for example, `\ifnum`; see Listings 27 and 28 for a before-and-after demonstration.

LISTING 27: if-else-fi construct before

```
\ifnum\radius>5
\ifnum\radius<16
\draw[decorate,...
\fi
\fi
```

LISTING 28: if-else-fi construct after

```
\ifnum\radius>5
    \ifnum\radius<16
      \draw[decorate,...
  \fi
\fi
```

`fileExtensionPreference` `latexindent.pl` can be called to act on a file without specifying the file extension. For example we can call `latexindent.pl myfile` in which case the script will look for `myfile` with the extensions specified in `fileExtensionPreference` in their numeric order. If no match is found, the script will exit. As with all of the fields, you should change and/or add to this as necessary.

LISTING 29: fileExtensionPreference

```
1  fileExtensionPreference:
2      .tex: 1
3      .sty: 2
4      .cls: 3
5      .bib: 4
```

Calling `latexindent.pl myfile` with the details specified in Listing 29 means that the script will first look for `myfile.tex`, then `myfile.sty`, `myfile.cls`, and finally `myfile.bib` in order.

`logFilePreferences` `latexindent.pl` writes information to `indent.log`, some of which can be customised by changing `logFilePreferences`; see Listing 30.

LISTING 30: logFilePreferences

```
1  logFilePreferences:
2      showEveryYamlRead: 1
3      showAlmagamatedSettings: 0
4      endLogFileWith: '--------------'
```

```
5       traceModeIncreaseIndent: '>>'
6       traceModeAddCurrentIndent: '||'
7       traceModeDecreaseIndent: '<<'
8       traceModeBetweenLines: "\n"
```

If you load your own user settings (see Section 4.2 on page 16) then `latexindent.pl` will detail them in `indent.log`; you can choose not to have the details logged by switching `showEveryYamlRead` to 0. Once all of your settings have been loaded, you can see the amalgamated settings by switching `showAlmagamatedSettings` to 1, if you wish. The log file will end with the characters given in `endLogFileWith`.

When `trace` mode is active (see page 5) verbose information is written to `indent.log`. The decoration of this information can be customised through the remaining fields given in Listing 30; note, in particular, the use of `"\n"` for escaped characters (using single quotes will not produce the same results).

`fileContentsEnvironments` `latexindent.pl` determines when the main document begins by looking for `\begin{document}`; it will not do so when inside any of the environments specified in `fileContentsEnvironments`, see Listing 31.

LISTING 31: `fileContentsEnvironments`

```
1  fileContentsEnvironments:
2      filecontents: 1
3      filecontents*: 1
```

The following fields are marked in red, as they are not necessary unless you wish to micro-manage your indentation scheme. Note that in each case, you should not *use the backslash.*

`checkunmatched` Assuming you keep `alwaysLookforSplitBraces` set to 1 (which is the default) then you don't need to worry about `checkunmatched`.

Should you wish to deactivate `alwaysLookforSplitBraces` by setting it to 0, then you can populate `checkunmatched` with commands that can split braces across lines–see Listing 32.

LISTING 32: `checkunmatched`

```
1  checkunmatched:
2      parbox: 1
3      vbox: 1
```

`checkunmatchedELSE` Similarly, assuming you keep `alwaysLookforSplitBraces` set to 1 (which is the default) then you don't need to worry about `checkunmatchedELSE`.

As in `checkunmatched`, should you wish to deactivate `alwaysLookforSplitBraces` by setting it to 0, then you can populate `checkunmatchedELSE` with commands that can split braces across lines *and* have an 'else' statement–see Listing 33.

LISTING 33: `checkunmatchedELSE`

```
1  checkunmatchedELSE:
2      pgfkeysifdefined: 1
3      DTLforeach: 1
4      ifthenelse: 1
```

checkunmatchedbracket Assuming you keep `alwaysLookforSplitBrackets` set to 1 (which is the default) then you don't need to worry about `checkunmatchedbracket`.

Should you wish to deactivate `alwaysLookforSplitBrackets` by setting it to 0, then you can populate `checkunmatchedbracket` with commands that can split *brackets* across lines– see Listing 34.

LISTING 34: `checkunmatchedbracket`

```
1  checkunmatchedbracket:
2      psSolid: 1
3      pgfplotstablecreatecol: 1
4      pgfplotstablesave: 1
5      pgfplotstabletypeset: 1
6      mycommand: 1
```

4.1.1 Hierarchy of fields

After reading the previous section, it should sound reasonable that `noAdditionalIndent`, `indentRules`, and `verbatim` all serve mutually exclusive tasks. Naturally, you may well wonder what happens if you choose to ask `latexindent.pl` to prioritize one above the other.

For example, let's say that (after reading Section 4.2) you put the fields in Listing 35 into one of your settings files.

LISTING 35: Conflicting ideas

```
1  indentRules:
2     myenvironment: "\t\t"
3  noAdditionalIndent:
4     myenvironment: 1
```

Clearly these fields conflict: first of all you are telling `latexindent.pl` that `myenvironment` should receive two tabs of indentation, and then you are telling it not to put any indentation in the environment. `latexindent.pl` will always make the decision to prioritize `noAdditionalIndent` above `indentRules` regardless of the order that you load them in your settings file. The first time it encounters `myenvironment` it will put a warning in `indent.log` and delete the offending key from `indentRules` so that any future conflicts will not have to be addressed.

Let's consider another conflicting example in Listing 36

LISTING 36: More conflicting ideas

```
1  lookForAlignDelims:
2     myenvironment: 1
3  verbatimEnvironments:
4     myenvironment: 1
```

This is quite a significant conflict–we are first of all telling `latexindent.pl` to look for alignment delimiters in `myenvironment` and then telling it that actually we would like `myenvironment` to be considered as a `verbatim`-like environment. Regardless of the order that we state Listing 36 the `verbatim` instruction will always win. As in Listing 35

you will only receive a warning in `indent.log` the first time `latexindent.pl` encounters `myenvironment` as the offending key is deleted from `lookForAlignDelims`.

To summarize, `latexindent.pl` will prioritize the various fields in the following order:

1. `verbatimEnvironments`
2. `noAdditionalIndent`
3. `indentRules`

4.2 `indentconfig.yaml` and `.indentconfig.yaml` (for user settings)

Editing `defaultSettings.yaml` is not ideal as it may be overwritten when updating your distribution–a better way to customize the settings to your liking is to set up your own settings file, `mysettings.yaml` (or any name you like, provided it ends with `.yaml`). The only thing you have to do is tell `latexindent.pl` where to find it.

`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 [**jacobo-diaz-hidden-config**] for providing this feature. In what follows, we will use `indentconfig.yaml`, but it is understood that this equally represents `.indentconfig.yaml` as well. If you have both files in existence, `indentconfig.yaml` takes priority.

For Mac and Linux users, their home directory is `~/username` while Windows (Vista onwards) is `C:\Users\username` [1] Listing 37 shows a sample `indentconfig.yaml` file.

LISTING 37: `indentconfig.yaml` (sample)

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

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

Note that the `.yaml` files you specify in `indentconfig.yaml` will be loaded in the order that you write them in. 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` feel free to start changing the switches and adding more environments to it as you see fit–have a look at Listing 38 for an example that uses four tabs for

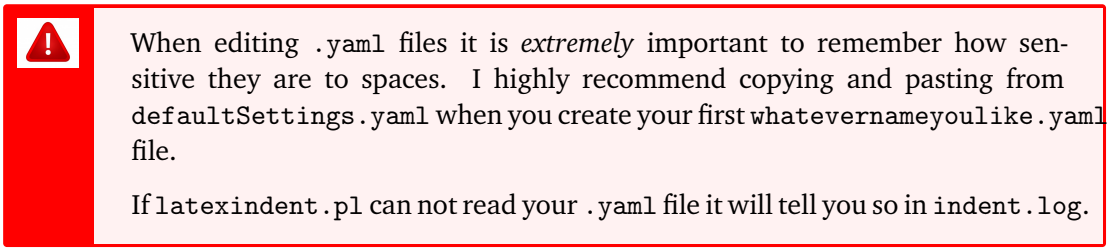
[1] If you're not sure where to put `indentconfig.yaml`, don't worry `latexindent.pl` will tell you in the log file exactly where to put it assuming it doesn't exist already.

the default indent, adds the `tabbing` environment to the list of environments that contains alignment delimiters, and adds the changes we described on page 13.

LISTING 38: `mysettings.yaml` (example)

```
1  # Default value of indentation
2  defaultIndent: "\t\t\t\t"
3
4  # environments that have tab delimiters, add more
5  # as needed
6  lookForAlignDelims:
7     tabbing: 1
8
9  # If you use the exam documentclass, you might
10 # like the following settings
11 # environments that have \item commands
12 indentAfterItems:
13     parts: 1
14
15 # commands to be treated like \item
16 itemNames:
17     part: 1
```

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 recognize then you'll get a warning, otherwise you'll get confirmation that `latexindent.pl` has read your settings file [2].



When editing `.yaml` files it is *extremely* important to remember how sensitive they are to spaces. I highly recommend copying and pasting from `defaultSettings.yaml` when you create your first `whatevernameyoulike.yaml` file.

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

4.3 `localSettings.yaml`

You may remember on page 6 we discussed the `-l` switch that tells `latexindent.pl` to look for `localSettings.yaml` in the *same directory* as `myfile.tex`. This settings file will be read *after* `defaultSettings.yaml` and, assuming they exist, user settings.

The *local* settings file may be called `localSettings.yaml`, and it can contain any switches that you'd like to change–a sample is shown in Listing 39.

LISTING 39: `localSettings.yaml` (example)

```
1  # Default value of indentation
2  defaultIndent: " "
3
4  # environments that have tab delimiters, add more
5  # as needed
6  lookForAlignDelims:
7     tabbing: 0
8
```

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

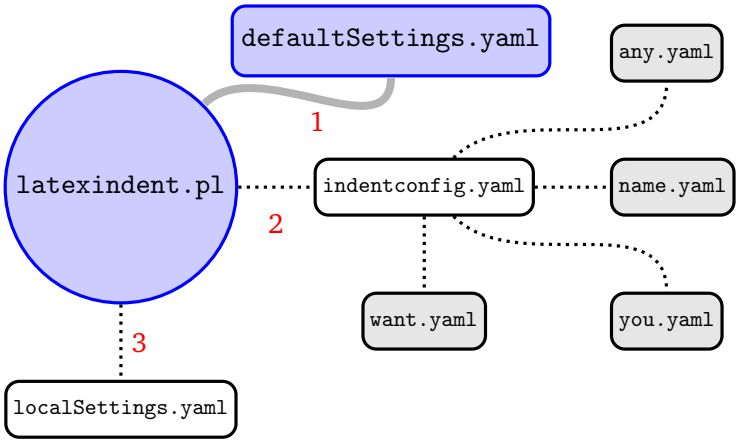


FIGURE 1: Schematic of the load order described in Section 4.4; solid lines represent mandatory files, dotted lines represent optional files. `indentconfig.yaml` can contain as many files as you like–the files will be loaded in order; if you specify settings for the same field in more than one file, the most recent takes priority.

```
 9  #  verbatim environments- environments specified
10  #  in this hash table will not be changed at all!
11  verbatimEnvironments:
12      cmhenvironment: 0
```

You can make sure that your local settings are loaded by checking `indent.log` for details–if `localSettings.yaml` can not be read then you will get a warning, otherwise you'll get confirmation that `latexindent.pl` has read `localSettings.yaml`.

If you'd prefer to name your `localSettings.yaml` file something different, (say, `myyaml.yaml`) then you can call `latexindent.pl` using, for example, `latexindent.pl -l=myyaml.yaml myfile.tex`.

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 4.3).

A visual representation of this is given in Figure 1.

4.5 An important example

I was working on a document that had the text shown in Listing 40.

LISTING 40: When to set `alwaysLookforSplitBrackets=0`

```
Hence determine how many zeros the function $h(x)=f(x)-g(x)$
has on the interval $[0,9)$.
\begin{shortsolution}
  The function $h$ has $10$ zeros on the interval $[0,9)$.
```

```
\end{shortsolution}
```

I had allowed `alwaysLookforSplitBrackets=1`, which is the default setting. Unfortunately, this caused undesired results, as `latexindent.pl` thought that the opening `[` in the interval notation (lines 2 and 4) was an opening brace that needed to be closed (with a corresponding `]`). Clearly this was inappropriate, but also expected since `latexindent.pl` was simply following its matching rules.

In this particular instance, I set up `localSettings.yaml` to contain `alwaysLookforSplitBrackets: 0` and then specified the commands that could split brackets across lines (such as `begin{axis}`) individually in `checkunmatchedbracket`. Another option would have been to wrap the the line in an environment from `noIndentBlock` which treats its contents as a verbatim environment.

5 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 of environments that contain delimiters–in other words, environments that are entered in `lookForAlignDelims`. Indeed, this is the only part of the script that can *potentially* remove lines from `myfile.tex`. Note that `indent.log` will always finish with a comparison of line counts before and after.

The routine works well for 'standard' blocks of code that have the same number of `&` per line, but it will not do anything for lines that do not–such examples include `tabular` environments that use `\multicolumn` or perhaps spread cell contents across multiple lines. For each alignment block (`tabular`, `align`, etc) `latexindent.pl` first of all makes a record of the maximum number of `&`; if each row does not have that number of `&` then it will not try to format that row. Details will be given in `indent.log` assuming that `trace` mode is active.

If you have a `verbatim`-like environment inside a `tabular`-like environment, the `verbatim` environment *will* be formatted, which is probably not what you want. I hope to address this in future versions, but for the moment wrap it in a `noIndentBlock` (see page 11).

You can run `latexindent` on `.sty`, `.cls` and any filetypes that you specify in `fileExtensionPreference` (see Listing 29 on page 13); if you find a case in which the script struggles, please feel free to report it at [**latexindent-home**], and in the meantime, consider using a `noIndentBlock` (see page 11).

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 [**latexindent-home**]; otherwise, feel free to find me on the `http://tex.stackexchange.com` site; I'm often around and in the chat room.

6 References

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 41 (`perl helloworld.pl`) then you will be able to run `latexindent.pl`, otherwise you may need to install the missing modules.

LISTING 41: helloworld.pl

```
#!/usr/bin/perl

use strict;
use warnings;
use FindBin;
use YAML::Tiny;
use File::Copy;
use File::Basename;
use Getopt::Long;
use File::HomeDir;
use Data::UUID;

print "hello world";
exit;
```

My default installation on Ubuntu 12.04 did *not* come with all of these modules as standard, but Strawberry Perl for Windows [**strawberryperl**] did.

Installing the modules given in Listing 41 will vary depending on your operating system and `Perl` distribution. For example, Ubuntu users might visit the software center, or else run

```
sudo perl -MCPAN -e 'install "File::HomeDir"'
```

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

```
sudo apt-get install perlbrew
perlbrew install perl-5.20.1
perlbrew switch perl-5.20.1
sudo apt-get install curl
curl -L http://cpanmin.us | perl - App::cpanminus
cpanm YAML::Tiny
cpanm File::HomeDir
use Data::UUID;
```

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

As of Version 2.1, `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 `latexindent.exe -t myfile.tex`.

B The `arara` rule

The `arara` rule (`indent.yaml`) contains lines such as those given in Listing 42. With this setup, the user *always* has to specify whether or not they want (in this example) to use the `trace` identifier.

LISTING 42: The `arara` rule

```
...
arguments:
- identifier: trace
```

```
    flag: <arara> @{ isTrue( parameters.trace, "-t" ) }
...
```

If you would like to have the `trace` option on by default every time you call `latexindent.pl` from `arara` (without having to write `% arara: indent: {trace: yes}`), then simply amend Listing 42 so that it looks like Listing 43.

LISTING 43: The `arara` rule (modified)

```
...
arguments:
- identifier: trace
  flag: <arara> @{ isTrue( parameters.trace, "-t" ) }
  default: "-t"
...
```

With this modification in place, you now simply to write `% arara: indent` and `trace` mode will be activated by default. If you wish to turn off `trace` mode then you can write `% arara: indent: {trace: off}`.

Of course, you can apply these types of modifications to *any* of the identifiers, but proceed with caution if you intend to do this for `overwrite`.

C Updating the `path` variable

`latexindent.pl` ships with a few scripts that can update the `path` variables [3]. If you're on a Linux or Mac machine, then you'll want `CMakeLists.txt` from [**latexindent-home**].

C.1 Add to path for Linux

To add `latexindent.pl` to the path for Linux, follow these steps:

1. download `latexindent.pl`, `defaultSettings.yaml`, to your chosen directory from [**latexindent-home**] ;
2. within your directory, create a directory called `path-helper-files` and download `CMakeLists.txt` and `cmake_uninstall.cmake.in` from [**latexindent-home**]/path-helper-files to this directory;
3. run `ls /usr/local/bin` to see what is *currently* in there;
4. run the commands given in Listing 44;
5. run `ls /usr/local/bin` again to check that `latexindent.pl` and `defaultSettings.yaml` have been added.

LISTING 44: Add to path from a Linux terminal

```
sudo apt-get install cmake
sudo apt-get update && sudo apt-get install build-essential
mkdir build && cd build
cmake ../path-helper-files
sudo make install
```

[3] Thanks to [**jasjuang**] for this feature!

To *remove* the files, run `sudo make uninstall`.

C.2 Add to path for Windows

To add `latexindent.exe` to the path for Windows, follow these steps:

1. download `latexindent.exe`, `defaultSettings.yaml`, `add-to-path.bat` from [**latexindent-home**] to your chosen directory;
2. open a command prompt and run `echo %path%` to see what is *currently* in your `%path%` variable;
3. right click on `add-to-path.bat` and *Run as administrator*;
4. log out, and log back in;
5. open a command prompt and run `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.

D Differences from Version 2.1 to 3.0 (and beyond)

There are a few (small) changes to the interface when comparing Version 2.1 to Version 3.0. Explicitly, these are:

OLD `latexindent.pl -o myfile.tex outputfile.tex`

NEW `latexindent.pl -o=outputfile.tex myfile.tex`

`latexindent.pl -o outputfile.tex myfile.tex`