indent.plx

Chris Hughes

March 31, 2013

Abstract

indent.plx is a Perl script that indents .tex files according to an indentation scheme that the user can modify to suit their taste. Environments, including those with alignment delimiters (such as tabular), commands, including those that can split braces and brackets across lines, are *usually* handled correctly by the script.

Contents

Listings

1 Before we begin

I first created `indent.plx` for helping me to format chapter files in a big project. After I blogged about it on the TEX stack exchange [?] 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! FIX

The `yaml`-based interface of `indent.plx` 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 `indent.plx`, but the release of `arara` has meant there is no need. Thank you to Paulo for all of your advice and encouragement.

`indent.plx` 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 4) 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 `indent.plx` 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.

2 Demonstration: before and after

3 How to use the script

There are two ways to use `indent.plx`: from the command line, and using `arara`. We will discuss how to change the settings and behaviour of the script in Section 4.

3.1 From the command line

`indent.plx` has a number of different switches/flags/options, which can be combined in any way that you like. `indent.plx` 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.

```
indent.plx myfile.tex
```

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

```
-w indent.plx -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 `indent.plx` 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.

`-o` `indent.plx -o myfile.tex outputfile.tex`

This will indent `myfile.tex` and output it to `outputfile.tex`, overwriting it (`outputfile.tex`) if it already exists. Note that if `indent.plx` 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 `indent.plx myfile.tex > outputfile.tex`

`-s` `indent.plx -s myfile.tex`

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

`-t` `indent.plx -t myfile.tex`

Tracing mode: verbose output will be given to `indent.log`. This is useful if `indent.plx` has made a mistake and you're trying to find out where and why.

`-l` `indent.plx -l myfile.tex`

Local settings: you might like to read Section 4 before using this switch. `indent.plx` 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.

3.2 From `arara`

Using `indent.plx` from the command line is fine for some folks, but others may find it easier to use from `arara`. `indent.plx` ships with an `arara` rule, `indent.yaml`, which you can either copy it to the directory of your other `arara` rules, or otherwise add the `indent.plx` directory to your `araraconfig.yaml` file.

Once you have told `arara` where to find your `indent` rule, you can use it any of the following ways (or combinations thereof).

LISTING 1: `arara` samples

```
% arara: indent
% arara: indent: {overwrite: yes}
% arara: indent: {output: myfile.tex}
% arara: indent: {silent: yes}
% arara: indent: {trace: yes}
% arara: indent: {localSettings: yes}
\documentclass{article}
...
```

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

Table 1: arara orb tags and corresponding switches

arara orb tags	switch
overwrite	-w
output	-o
silent	-s
trace	-t
localSettings	-l

4 default, user, and local settings

`indent.plx` 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 `indent.plx`. The code 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 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 6).

`backupExtension` `.bak`

If you call `indent.plx` 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 `indent.plx` 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 `indent.plx` (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`.

`indentPreamble` `0`

The preamble of a document can sometimes contain some trickier code for `indent↪.plx` to work with. By default, `indent.plx` 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 `indent.plx` to look for commands that can split *braces* across lines, such as `parbox`, `tikzset`, etc. In older versions of `indent.plx` 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.

`alwaysLookforSplitBrackets` 1

This switch tells `indent.plx` to look for commands that can split *brackets* across lines, such as `psSolid`, `pgfplotstabletypeset`, etc. In older versions of `indent↪.plx` 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.

`indentAfterDocument` 0

This switch tells `indent.plx` to indent after `\end{document}` or not.

`lookForAlignDelims` This is the first example of a field in `defaultSettings.yaml` that has more than one line; listing 2 shows more details.

LISTING 2: `lookForAlignDelims`

```
lookForAlignDelims:
   tabular: 1
   align: 1
   align*: 1
   alignat: 1
   alignat*: 1
   cases: 1
   dcases: 1
   aligned: 1
   pmatrix: 1
   listabla: 1
```

You can populate this field with any other environments that you have that contain `&`. If you change your mind, just turn them off by setting them to `0` instead.

`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 3.

LISTING 3: `verbatimEnvironments`

```
verbatimEnvironments:
    verbatim: 1
    lstlisting: 1
```

`noIndentBlock` If you have a block of code that you don't want `indent.plx` 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 4.

Listing 4: `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 5 for example.

Listing 5: `noIndentBlock` demonstration

```
% \begin{noindent}
          this code
                  won't
      be touched
                           by
                 \lstinline!indent.plx!
%\end{noindent}
```

`noAdditionalIndent` If you would prefer some of your environments or commands not to receive any additional indent, then populate `noAdditionalIndent`; see listing 6. Note that these environments will still receive the *current* level of indentation unless they belong to `verbatimEnvironments`, or `noIndentBlock`.

Listing 6: `noAdditionalIndent`

```
noAdditionalIndent:
    document: 1
    pccexample: 1
    pccdefinition: 1
    problem: 1
    exercises: 1
    pccsolution: 1
    foreach: 0
    widepage: 1
    comment: 1
    \[: 0
    frame: 0
```

`indentRules` If you would prefer to specify individual rules for certain environments or commands, just populate `indentRules`; see listing 7

Listing 7: `indentRules`

```
indentRules:
   myenvironment: "\t\t"
   anotherenvironment: "\t\t\t\t"
   \[: "\t"
```

!!! *The following fields are marked in red, as they are not necessary unless you wish to micro manage your indentation scheme.*

`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 8.

Listing 8: `checkunmatched`

```
checkunmatched:
    parbox: 1
    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 9.

Listing 9: `checkunmatchedELSE`

```
checkunmatchedELSE:
    pgfkeysifdefined: 1
    DTLforeach: 1
    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 10.

Listing 10: `checkunmatchedbracket`

```
checkunmatchedbracket:
    psSolid: 1
    pgfplotstablecreatecol: 1
    pgfplotstablesave: 1
    pgfplotstabletypeset: 1
    mycommand: 1
```

4.2 `indentconfig.yaml` (for user settings)

A better way to change the settings 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 `indent.plx` where to find it.

`indent.plx` will always check your home directory for `indentconfig.yaml`, which is a plain text file you can create that contains the *absolute* paths for any settings files that you wish `indent.plx` to load– see listing 11 for a sample.

Listing 11: `indentconfig.yaml`

```
# Paths to user settings for indent.plx
#
# 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
```

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 your 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 12 for an example that uses four tabs for the default indent, and adds the `tabbing` environment to the list of environments that contains alignment delimiters.

Listing 12: `mysettings.yaml` (example)

```
# Default value of indentation
defaultIndent: "\t\t\t\t"

# environments that have tab delimiters, add more
# as needed
lookForAlignDelims:
   tabbing: 1
```

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

FIX

4.3 localSettings.yaml

You may remember on page 3 we discussed the `-l` switch that tells `indent.plx` 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.

In contrast to the *user* settings which can be named anything you like (provided that they are detailed in `indentconfig.yaml`), the *local* settings file must be called `localSettings.yaml`. It can contain any switches that you'd like to change– a sample is shown in listing 13.

Listing 13: `localSettings.yaml` (example)

```
# Default value of indentation
defaultIndent: " "

# environments that have tab delimiters, add more
# as needed
lookForAlignDelims:
   tabbing: 0

```

```
#  verbatim environments- environments specified
#  in this hash table will not be changed at all!
verbatimEnvironments:
    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 `indent.plx` has read `localSettings.yaml`.

4.4 Settings load order

`indent.plx` loads the settings files in the following order:

1. `defaultSettings.yaml` (always loaded, can not be renamed)
2. `anyUserSettings.yaml` (and any other arbitrarily-named files specified in `indentconfig`↪ `.yaml`)
3. `localSettings.yaml` (if found in same directory as `myfile.tex` and called with `-l` switch; can not be renamed)

5 Known limitations

nested align delimiter blocks tables with multicolumn