`latexindent.pl`

Version 3.0

Chris Hughes

February 13, 2017

`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 modifiy line breaks, and add comment symbols. All user options are customisable via the switches in the YAML interface.

Contents

0.1 Commands and the strings between their arguments 1

Listings

Listing 1: `commandCodeBlocks` · · · · · · · · · · 1
Listing 2: `pstricks1.tex` · · · · · · · · · · 2
Listing 3: `pstricks1` default output · · · · · · · · · · 2
Listing 4: `pstricks1.tex` using Listing 5 · · · · · · · · · · 2
Listing 5: `noRoundParentheses.yaml` · · · · · · · · · · 2
Listing 6: `pstricks1.tex` using Listing 7 · · · · · · · · · · 2
Listing 7: `defFunction.yaml` · · · · · · · · · · 2
Listing 8: `tikz-node1.tex` · · · · · · · · · · 3
Listing 9: `tikz-node1` default output · · · · · · · · · · 3
Listing 10: `tikz-node1.tex` using Listing 11 · · · · · · · · · · 3
Listing 11: `draw.yaml` · · · · · · · · · · 3
Listing 12: `tikz-node1.tex` using Listing 13 · · · · · · · · · · 3
Listing 13: `no-to.yaml` · · · · · · · · · · 3

0.1 Commands and the strings between their arguments

The `command` code blocks will always look for optional (square bracketed) and mandatory (curly braced) arguments; there are switches that can allow them to contain other strings.

`commandCodeBlocks`: ⟨*fields*⟩

The `commandCodeBlocks` field contains a few switches detailed in Listing 1.

Listing 1: `commandCodeBlocks`

```
commandCodeBlocks:
    roundParenthesesAllowed: 1
    stringsAllowedBetweenArguments:
        node: 1
        at: 1
        to: 1
        decoration: 1
        ++: 1
        --: 1
```

and contributors! (See **??** on page ??.) For all communication, please visit [6].

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 2.

LISTING 2: pstricks1.tex

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

LISTING 3: pstricks1 default output

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

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

By default, because roundParenthesesAllowed is set to 1 in Listing 1, then latexindent.pl will allow round parenthesis between optional and mandatory arguments. In the case of the code in Listing 2, latexindent.pl finds *all* the arguments appropriately.

The default output from running latexindent.pl on Listing 2 actually leaves it unchanged (see Listing 3). Upon using the YAML settings in Listing 5, and running the command

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

we obtain the output given in Listing 4.

LISTING 4: pstricks1.tex using Listing 5

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

LISTING 5: noRoundParentheses.yaml

```
commandCodeBlocks:
    roundParenthesesAllowed: 0
```

Notice the difference between Listing 3 and Listing 4; in particular, in Listing 4, 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 **??** on page ??) which, by default, assume indentation for their body, and hence the tabbed indentation in Listing 4.

Let's explore this using the YAML given in Listing 7 and run the command

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

then the output is as in Listing 6.

LISTING 6: pstricks1.tex using Listing 7

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

LISTING 7: defFunction.yaml

```
indentRules:
    defFunction:
        body: " "
```

Notice in Listing 6 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 7.

`stringsAllowedBetweenArguments`: ⟨*fields*⟩

`tikz` users may well specify code such as that given in Listing 8; processing this code using `latexindent.pl` gives the default output in Listing 9.

LISTING 8: `tikz-node1.tex`

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

LISTING 9: `tikz-node1` default output

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

With reference to Listing 1 on page 1, we see that the strings

to, node, ++

are all allowed to appear between arguments, as they are each set to 1. This means that when `latexindent.pl` processes Listing 8, 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 11 and running the command

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

we receive the output given in Listing 10.

LISTING 10: `tikz-node1.tex` using Listing 11

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

LISTING 11: `draw.yaml`

```
indentRules:
    draw:
        body: "  "
```

Notice that each line after the `\draw` command (its 'body') in Listing 10 has been given the appropriate two-spaces worth of indentation specified in Listing 11.

Let's compare this with the output from using the YAML settings in Listing 13, and running the command

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

given in Listing 12.

LISTING 12: tikz-node1.tex using Listing 13

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

LISTING 13: no-to.yaml

```
commandCodeBlocks:
    stringsAllowedBetweenArguments:
        to: 0
```

In this case, `latexindent.pl` sees that

- the `\draw` command finishes after the `(c)` as (`stringsAllowedBetweenArguments` has `to` set to 0)
- it finds a `namedGroupingBracesBrackets` called `to` (see **??** on page ??) *with* argument `[in=110,out=-90]`
- it finds another `namedGroupingBracesBrackets` but this time called `node` with argument `[below,align=left,scale=0.5]`