latexindent.pl

Version 3.0

Chris Hughes

February 1, 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

Listings

0.1 Conflicting poly-switches: sequential code blocks

It is very easy to have conflicting poly-switches; if we use the example from **??** on page ??, and consider the YAML settings given in Listing 2. The output from running

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

is given in Listing 2.

Listing 1: mycommand1.tex using Listing 2

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

Listing 2: mycom-mlb4.yaml

```
modifyLineBreaks:
    mandatoryArguments:
        LCuBStartsOnOwnLine: -1
        RCuBFinishesWithLineBreak: 1
```

Studying Listing 2, we see that the two poly-switches are at opposition with one another:

and contributors! (See **??** on page ??.) For all communication, please visit [1].

- 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 1, 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 prioirty.

We can explore this further by considering the YAML settings in Listing 4; upon runnning the command

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

we obtain the output given in Listing 3.

LISTING 3: `mycommand1.tex` using Listing 4

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

LISTING 4: `mycom-mlb5.yaml`

```
modifyLineBreaks:
    mandatoryArguments:
        LCuBStartsOnOwnLine: 1
        RCuBFinishesWithLineBreak: -1
```

As previously, the most-recently-processed code block takes prioirty – as before, the second (i.e, *last*) argument. Exploring this further, we consider the YAML settings in Listing 6, which give associated output in Listing 5.

LISTING 5: `mycommand1.tex` using Listing 6

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

LISTING 6: `mycom-mlb6.yaml`

```
modifyLineBreaks:
    mandatoryArguments:
        LCuBStartsOnOwnLine: 2
        RCuBFinishesWithLineBreak: -1
```

Note that a `%` *has* been added to the trailing first `}`; this is because:

- while processing the *first* argument, the trailing line break has been removed (`RCuBFinishesWithLineBreak` set to −1);
- while processing the *second* argument, `latexindent.pl` finds that it does *not* begin on its own line, and so because `LCuBStartsOnOwnLine` is set to 2, it adds a comment, followed by a line break.

0.2 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 7, noting that it contains nested environments.

LISTING 7: `nested-env.tex`

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

Let's use the YAML settings given in Listing 9, which upon running the command

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

gives the output in Listing 9.

Listing 8: nested-env.tex using Listing 9

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

Listing 9: nested-env-mlb1.yaml

```
modifyLineBreaks:
    environments:
        EndStartsOnOwnLine: -1
        EndFinishesWithLineBreak: 1
```

In Listing 9, let's first of all note that both environments have received the appropriate (default) indentation; secondly, note that the poly-switch `EndStartsOnOwnLine` appears to have won the conflict, as `\end{one}` has had its leading line break removed.

To understand it, let's talk about the three basic phases of `latexindent.pl`:

1. Phase 1: packing, in which code blocks are replaced with unique ids, working from *the inside to the outside* – for example, in Listing 7, 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 (`BodyStartsOnOwnLine:` 1/2) or removed (`BodyStartsOnOwnLine:` −1);
   - line breaks at the end of the body can be added (see `EndStartsOnOwnLine` 1/2) or removed (see `EndStartsOnOwnLine:` −1);
2. Phase 2: indentation, in which white space is added to the begin, body, and end statements;
3. Phase 3: unpacking, in which unique ids are replaced by their *indented* code blocks; if the -m switch is active, then during this phase,
   - line breaks before *begin* statements can be added or removed (if `BeginStartsOnOwnLine:` −1);
   - line breaks after *end* statements can be removed but *NOT* added (see `EndFinishesWithLineBreak`).

With reference to Listing 9, this means that during Phase 1, the line break ahead of both of the `\end` statements has been removed, first of all by the `two` environment, and then the `one` environment.

We can explore this further using the poly-switches in Listing 11, which give the output in Listing 11.

Listing 10: nested-env.tex using Listing 11

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

Listing 11: nested-env-mlb2.yaml

```
modifyLineBreaks:
    environments:
        EndStartsOnOwnLine: 1
        EndFinishesWithLineBreak: -1
```

During phase

FIX