latexindent.pl

Version 3.12

Chris Hughes *

2021-09-16

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 \texttt{tabular}), and commands, including those that can split braces and brackets across lines, are usually handled correctly by the script. Options for \texttt{verbatim}-like environments and commands, together with indentation after headings (such as \texttt{chapter}, \texttt{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 11.

---

Contents

1 Introduction 10
  1.1 Thanks .......................... 10
  1.2 License .......................... 10
  1.3 About this documentation ....... 10
  1.4 Quick start ........................ 11
  1.5 A word about regular expressions 12

2 Demonstration: before and after 13

3 How to use the script 14
  3.1 From the command line ............ 14
  3.2 From arara ........................ 20
  3.3 Summary of exit codes ............. 20

4 indentconfig.yaml, local settings and the -y switch 21
  4.1 indentconfig.yaml and .indentconfig.yaml 21
  4.2 localSettings.yaml and friends ....... 22
  4.3 The -y|yaml switch ............... 23
  4.4 Settings load order .............. 23

* and contributors! See Section 11.2 on page 138. For all communication, please visit [11].
CONTENTS

5 defaultSettings.yaml .......................... 25
  5.1 Backup and log file preferences ......................................... 25
  5.2 Verbatim code blocks .................................................. 27
  5.3 filecontents and preamble ............................................ 29
  5.4 Indentation and horizontal space ...................................... 30
  5.5 Aligning at delimiters .................................................. 30
    5.5.1 lookForAlignDelims: spacesBeforeAmpersand ......................... 35
    5.5.2 lookForAlignDelims: alignFinalDoubleBackSlash ....................... 36
    5.5.3 lookForAlignDelims: the dontMeasure feature ......................... 37
    5.5.4 lookForAlignDelims: the delimiterRegEx and delimiterJustification feature .... 39
  5.6 Indent after items, special headings ................................ 41
  5.7 The code blocks known latexindent.pl ................................ 48
  5.8 noAdditionalIndent and indentRules .................................. 48
    5.8.1 Environments and their arguments .................................. 50
    5.8.2 Environments with items ............................................ 56
    5.8.3 Commands with arguments ........................................... 57
    5.8.4 ifelsefi code blocks ................................................. 59
    5.8.5 specialBeginEnd code blocks ......................................... 61
    5.8.6 afterHeading code blocks ........................................... 62
    5.8.7 The remaining code blocks ........................................... 64
      5.8.7.1 keyEqualsValuesBracesBrackets ................................ 64
      5.8.7.2 namedGroupingBracesBrackets .................................... 64
      5.8.7.3 UnNamedGroupingBracesBrackets ................................ 65
      5.8.7.4 filecontents .................................................... 65
    5.8.8 Summary .............................................................. 65
  5.9 Commands and the strings between their arguments ................... 66

6 The -m (modifylinebreaks) switch ......................................... 71
  6.1 Text Wrapping ......................................................... 72
    6.1.1 Text wrap quick start .............................................. 73
    6.1.2 textWrapOptions: modifying line breaks by text wrapping ............ 73
    6.1.3 Text wrapping on a per-code-block basis ........................... 76
  6.2 removeParagraphLineBreaks: modifying line breaks for paragraphs ....... 82
  6.3 Combining removeParagraphLineBreaks and textWrapOptions ............... 88
    6.3.1 text wrapping beforeFindingChildCodeBlocks ........................ 89
  6.4 Summary of text wrapping ............................................. 91
  6.5 oneSentencePerLine: modifying line breaks for sentences ............... 91
    6.5.1 sentencesFollow .................................................... 93
    6.5.2 sentencesBeginWith ............................................... 94
    6.5.3 sentencesEndWith .................................................. 95
    6.5.4 Features of the oneSentencePerLine routine ........................ 97
    6.5.5 Text wrapping and indenting sentences ............................ 98
  6.6 Poly-switches .......................................................... 100
    6.6.1 Poly-switches for environments .................................... 100
      6.6.1.1 Adding line breaks: BeginStartsOnOwnLine and BodyStartsOnOwn-Line .......... 101
      6.6.1.2 Adding line breaks using EndStartsOnOwnLine and EndFinishesWith-LineBreak 103
      6.6.1.3 poly-switches 1, 2, and 3 only add line breaks when necessary .......... 104
      6.6.1.4 Removing line breaks (poly-switches set to −1) .......................... 105
      6.6.1.5 About trailing horizontal space .................................... 106
      6.6.1.6 poly-switch line break removal and blank lines ....................... 106
    6.6.2 Poly-switches for double back slash ................................ 108
      6.6.2.1 Double back slash starts on own line .................................. 108
      6.6.2.2 Double back slash finishes with line break .......................... 109
      6.6.2.3 Double back slash poly-switches for specialBeginEnd ................. 109
      6.6.2.4 Double back slash poly-switches for optional and mandatory arguments .. 110
      6.6.2.5 Double back slash optional square brackets ........................ 111
LISTING 9: \texttt{tikzset}.\texttt{tex} default output  
LISTING 10: \texttt{pstricks}.\texttt{tex}  
LISTING 11: \texttt{pstricks}.\texttt{tex} default output  
LISTING 14: The encoding option for \texttt{indentconfig}.\texttt{yaml}  
LISTING 16: fileExtensionPreference  
LISTING 17: logFilePreferences  
LISTING 18: verbatimEnvironments  
LISTING 19: \texttt{verbatimCommands}  
LISTING 20: noIndentBlock  
LISTING 21: noIndentBlock.\texttt{tex}  
LISTING 22: noIndentBlock1.\texttt{tex}  
LISTING 23: noindent1.\texttt{yaml}  
LISTING 24: noindent2.\texttt{yaml}  
LISTING 25: noindent3.\texttt{yaml}  
LISTING 26: noIndentBlock1.\texttt{tex} using Listing 23 or Listing 24  
LISTING 27: noIndentBlock1.\texttt{tex} using Listing 25  
LISTING 28: fileContentsEnvironments  
LISTING 29: lookForPreamble  
LISTING 30: Motivating preambleCommandsBeforeEnvironments  
LISTING 31: removeTrailingWhitespace  
LISTING 34: \texttt{tabular1}.\texttt{tex}  
LISTING 35: \texttt{tabular1}.\texttt{tex} default output  
LISTING 36: lookForAlignDelims (advanced)  
LISTING 37: \texttt{tabular2}.\texttt{tex}  
LISTING 38: \texttt{tabular2}.\texttt{yaml}  
LISTING 39: \texttt{tabular3}.\texttt{yaml}  
LISTING 40: \texttt{tabular4}.\texttt{yaml}  
LISTING 41: \texttt{tabular5}.\texttt{yaml}  
LISTING 42: \texttt{tabular6}.\texttt{yaml}  
LISTING 43: \texttt{tabular7}.\texttt{yaml}  
LISTING 44: \texttt{tabular8}.\texttt{yaml}  
LISTING 45: \texttt{tabular2}.\texttt{tex} default output  
LISTING 46: \texttt{tabular2}.\texttt{tex} using Listing 38  
LISTING 47: \texttt{tabular2}.\texttt{tex} using Listing 39  
LISTING 48: \texttt{tabular2}.\texttt{tex} using Listings 38 and 40  
LISTING 49: \texttt{tabular2}.\texttt{tex} using Listings 38 and 41  
LISTING 50: \texttt{tabular2}.\texttt{tex} using Listings 38 and 42  
LISTING 51: \texttt{tabular2}.\texttt{tex} using Listings 38 and 43  
LISTING 52: \texttt{tabular2}.\texttt{tex} using Listings 38 and 44  
LISTING 53: \texttt{aligned1}.\texttt{tex}  
LISTING 54: \texttt{aligned1-default}.\texttt{tex}  
LISTING 55: \texttt{sba1}.\texttt{yaml}  
LISTING 56: \texttt{sba2}.\texttt{yaml}  
LISTING 57: \texttt{sba3}.\texttt{yaml}  
LISTING 58: \texttt{sba4}.\texttt{yaml}  
LISTING 59: \texttt{aligned1-mod1}.\texttt{tex}  
LISTING 60: \texttt{sba5}.\texttt{yaml}  
LISTING 61: \texttt{sba6}.\texttt{yaml}  
LISTING 62: \texttt{aligned1-mod5}.\texttt{tex}  
LISTING 63: \texttt{aligned1}.\texttt{tex} using Listing 64  
LISTING 64: \texttt{sba7}.\texttt{yaml}  
LISTING 65: \texttt{tabular4}.\texttt{tex}  
LISTING 66: \texttt{tabular4}.\texttt{default}.\texttt{tex}  
LISTING 67: \texttt{tabular4}.\texttt{FDBS}.\texttt{tex}  
LISTING 68: \texttt{matrix1}.\texttt{tex}  
LISTING 69: \texttt{matrix1}.\texttt{tex} default output  
LISTING 70: align-block  
LISTING 71: align-block.\texttt{tex} default output  
LISTING 72: \texttt{tabular-DM}.\texttt{tex}  
LISTING 73: \texttt{tabular-DM}.\texttt{tex} default output  
LISTING 74: \texttt{tabular-DM}.\texttt{tex} using Listing 75  
LISTING 75: dontMeasure1.\texttt{yaml}  
LISTING 76: \texttt{tabular-DM}.\texttt{tex} using Listing 77 or Listing 79  
LISTING 77: dontMeasure2.\texttt{yaml}  
LISTING 78: \texttt{tabular-DM}.\texttt{tex} using Listing 79 or Listing 79  
LISTING 79: dontMeasure3.\texttt{yaml}  
LISTING 80: dontMeasure4.\texttt{yaml}  
LISTING 81: \texttt{tabular-DM}.\texttt{tex} using Listing 82  
LISTING 82: dontMeasure5.\texttt{yaml}  
LISTING 83: \texttt{tabular-DM}.\texttt{tex} using Listing 84  
LISTING 84: dontMeasure6.\texttt{yaml}  
LISTING 85: tabbing  
LISTING 86: tabbing.\texttt{tex} default output  
LISTING 87: tabbing.\texttt{tex} using Listing 88  
LISTING 88: delimiterRegEx1.\texttt{yaml}  
LISTING 89: \texttt{tabbing}.\texttt{tex} using Listing 90  
LISTING 90: delimiterRegEx2.\texttt{yaml}  
LISTING 91: \texttt{tabbing}.\texttt{tex} using Listing 92  
LISTING 92: delimiterRegEx3.\texttt{yaml}  
LISTING 93: tabbing1  
LISTING 94: tabbing1-mod4  
LISTING 95: delimiterRegEx4.\texttt{yaml}  
LISTING 96: tabbing1-mod5  
LISTING 97: delimiterRegEx5.\texttt{yaml}  
LISTING 98: indentAfterItems  
LISTING 99: items1  
LISTING 100: items1.\texttt{tex} default output  
LISTING 101: itemNames  
LISTING 102: specialBeginEnd  
LISTING 103: \texttt{special1}.\texttt{tex} before  
LISTING 104: \texttt{special1}.\texttt{tex} default output  
LISTING 105: \texttt{specialLR}.\texttt{tex}  
LISTING 106: \texttt{specialLeftRight}.\texttt{yaml}  
LISTING 107: \texttt{specialBeforeCommand}.\texttt{yaml}  
LISTING 108: \texttt{specialLR}.\texttt{tex} using Listing 106
Listing 109: specialLR.tex using Listings 106 and 107 ........................................... 43
Listing 110: special2.tex ......................................................... 43
Listing 111: middle.yaml ....................................................... 44
Listing 112: special2.tex using Listing 111 .......................... 44
Listing 113: middle1.yaml ..................................................... 44
Listing 114: special2.tex using Listing 113 .......................... 44
Listing 115: special-verb1.yaml ............................................. 44
Listing 116: special3.tex and output using Listing 115 ........... 44
Listing 117: special-align.tex ................................................. 45
Listing 118: edge-node1.yaml ................................................. 45
Listing 119: special-align.tex using Listing 118 ................. 45
Listing 120: edge-node2.yaml ............................................... 45
Listing 121: special-align.tex using Listing 120 ................. 45
Listing 122: indentAfterHeadings ......................................... 46
Listing 123: headings1.yaml ................................................. 46
Listing 124: headings1.tex ................................................... 46
Listing 125: headings1.tex using Listing 123 ..................... 47
Listing 126: headings1.tex second modification .................. 47
Listing 127: mult-nested.tex .................................................. 47
Listing 128: mult-nested.tex default output ....................... 47
Listing 129: max-indentation1.yaml ...................................... 48
Listing 130: mult-nested.tex using Listing 129 ................... 48
Listing 131: myenv.tex ......................................................... 50
Listing 132: myenv-noAdd1.yaml .......................................... 50
Listing 133: myenv-noAdd2.yaml .......................................... 50
Listing 134: myenv.tex output (using either Listing 132 or Listing 133) ............................................................... 50
Listing 135: myenv-noAdd3.yaml .......................................... 51
Listing 136: myenv-noAdd4.yaml .......................................... 51
Listing 137: myenv.tex output (using either Listing 135 or Listing 136) ............................................................... 51
Listing 138: myenv-args.tex .................................................. 51
Listing 139: myenv-args.tex using Listing 132 ................... 51
Listing 140: myenv-noAdd5.yaml .......................................... 52
Listing 141: myenv-noAdd6.yaml .......................................... 52
Listing 142: myenv-args.tex using Listing 140 ................... 52
Listing 143: myenv-args.tex using Listing 141 ................... 52
Listing 144: myenv-rules1.yaml ........................................... 52
Listing 145: myenv-rules2.yaml ........................................... 52
Listing 146: myenv.tex output (using either Listing 144 or Listing 145) ............................................................... 53
Listing 147: myenv-args.tex using Listing 144 ................... 53
Listing 148: myenv-rules3.yaml .......................................... 53
Listing 149: myenv-rules4.yaml .......................................... 53
Listing 150: myenv-args.tex using Listing 148 ................... 54
Listing 151: myenv-args.tex using Listing 149 ................... 54
Listing 152: noAdditionalIndentGlobal .................................. 54
Listing 153: myenv-args.tex using Listing 152 ................... 54
Listing 154: myenv-args.tex using Listings 144 and 152 ............................................................... 54
Listing 155: opt-args-no-add-glob.yaml ........................... 55
Listing 156: mand-args-no-add-glob.yaml ........................... 55
Listing 157: myenv-args.tex using Listing 155 ................... 55
Listing 158: myenv-args.tex using Listing 156 ................... 55
Listing 159: indentRulesGlobal ............................................. 55
Listing 160: myenv-args.tex using Listing 159 ................... 56
Listing 161: myenv-args.tex using Listings 144 and 159 ........ 56
Listing 162: opt-args-indent-rules-glob.yaml ........................ 56
Listing 163: mand-args-indent-rules-glob.yaml ................. 56
Listing 164: myenv-args.tex using Listing 162 ................... 56
Listing 165: myenv-args.tex using Listing 163 ................... 56
Listing 166: item-noAdd1.yaml ............................................ 56
Listing 167: item-rules1.yaml ............................................ 56
Listing 168: items1.tex using Listing 166 .......................... 57
Listing 169: items1.tex using Listing 167 ......................... 57
Listing 170: items-noAdditionalGlobal.yaml ........................ 57
Listing 171: items-indentRulesGlobal.yaml .......................... 57
Listing 172: mycommand.tex ................................................. 57
Listing 173: mycommand.tex default output ....................... 57
Listing 174: mycommand-noAdd1.yaml ................................... 58
Listing 175: mycommand-noAdd2.yaml ................................... 58
Listing 176: mycommand.tex using Listing 174 ................. 58
Listing 177: mycommand.tex using Listing 175 ................. 58
Listing 178: mycommand-noAdd3.yaml ................................... 58
Listing 179: mycommand-noAdd4.yaml ................................... 58
Listing 180: mycommand.tex using Listing 178 ................. 58
Listing 181: mycommand.tex using Listing 179 ................. 58
Listing 182: mycommand-noAdd5.yaml ................................... 59
Listing 183: mycommand-noAdd6.yaml ................................... 59
Listing 184: mycommand.tex using Listing 182 ................. 59
Listing 185: mycommand.tex using Listing 183 ................. 59
Listing 186: elseifi1.tex ..................................................... 59
Listing 187: elseifi1.tex default output ........................... 59
Listing 188: ifnum-noAdd.yaml ............................................ 59
Listing 189: ifnum-indent-rules.yaml ................................... 59
Listing 190: elseifi1.tex using Listing 188 ..................... 60
Listing 191: elseifi1.tex using Listing 189 ..................... 60
Listing 192: elseifi-noAdd-glob.yaml .................................. 60
Listing 193: elseifi-indent-rules-global.yaml ................... 60
Listing 194: elseifi1.tex using Listing 192 ..................... 60
Listing 195: elseifi1.tex using Listing 193 ..................... 60
Listing 196: elseifi2.tex ..................................................... 60
Listing 197: elseifi2.tex default output ........................... 60
Listing 198: displayMath-noAdd.yaml ................................... 61
Listing 199: displayMath-indent-rules.yaml ....................... 61
<table>
<thead>
<tr>
<th>Listing</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>298</td>
<td><code>shortlines-opt.tex</code></td>
<td>83</td>
</tr>
<tr>
<td>299</td>
<td><code>shortlines-mltex.tex</code></td>
<td>83</td>
</tr>
<tr>
<td>300</td>
<td><code>shortlines-opt1.tex</code></td>
<td>84</td>
</tr>
<tr>
<td>301</td>
<td><code>shortlines-envs.tex</code></td>
<td>84</td>
</tr>
<tr>
<td>302</td>
<td><code>remove-para2.yaml</code></td>
<td>84</td>
</tr>
<tr>
<td>303</td>
<td><code>remove-para3.yaml</code></td>
<td>84</td>
</tr>
<tr>
<td>304</td>
<td><code>shortlines-new2.tex</code></td>
<td>85</td>
</tr>
<tr>
<td>305</td>
<td><code>shortlines-envs3.tex</code></td>
<td>85</td>
</tr>
<tr>
<td>306</td>
<td><code>shortlines-md.tex</code></td>
<td>86</td>
</tr>
<tr>
<td>307</td>
<td><code>remove-para4.yaml</code></td>
<td>86</td>
</tr>
<tr>
<td>308</td>
<td><code>shortlines-md4.tex</code></td>
<td>86</td>
</tr>
<tr>
<td>309</td>
<td><code>paragraphsStopAt</code></td>
<td>87</td>
</tr>
<tr>
<td>310</td>
<td><code>sl-stop.txt</code></td>
<td>87</td>
</tr>
<tr>
<td>311</td>
<td><code>stop-command.yaml</code></td>
<td>87</td>
</tr>
<tr>
<td>312</td>
<td><code>stop-comment.yaml</code></td>
<td>87</td>
</tr>
<tr>
<td>313</td>
<td><code>sl-stop4.txt</code></td>
<td>87</td>
</tr>
<tr>
<td>314</td>
<td><code>sl-stop4-command.txt</code></td>
<td>88</td>
</tr>
<tr>
<td>315</td>
<td><code>sl-stop4-comment.txt</code></td>
<td>88</td>
</tr>
<tr>
<td>316</td>
<td><code>textwrap7.txt</code></td>
<td>88</td>
</tr>
<tr>
<td>317</td>
<td><code>textwrap7.txt using Listing 275</code></td>
<td>88</td>
</tr>
<tr>
<td>318</td>
<td><code>textwrap7-mod12.tex</code></td>
<td>89</td>
</tr>
<tr>
<td>319</td>
<td><code>textwrap12.yaml</code></td>
<td>89</td>
</tr>
<tr>
<td>320</td>
<td><code>textwrap-bfccb.tex</code></td>
<td>89</td>
</tr>
<tr>
<td>321</td>
<td><code>textwrap-bfccb-mod12.tex</code></td>
<td>89</td>
</tr>
<tr>
<td>322</td>
<td><code>textwrap13.yaml</code> (tweaked quick start)</td>
<td>90</td>
</tr>
<tr>
<td>323</td>
<td><code>textwrap-bfccb-mod13.tex</code></td>
<td>90</td>
</tr>
<tr>
<td>324</td>
<td><code>textwrap-bfccb-mod14.txt</code></td>
<td>90</td>
</tr>
<tr>
<td>325</td>
<td><code>textwrap14.yaml</code></td>
<td>90</td>
</tr>
<tr>
<td>326</td>
<td><code>textwrap15.yaml</code></td>
<td>91</td>
</tr>
<tr>
<td>327</td>
<td><code>onesentenceperline</code></td>
<td>92</td>
</tr>
<tr>
<td>328</td>
<td><code>multiple-sentences.tex</code></td>
<td>92</td>
</tr>
<tr>
<td>329</td>
<td><code>multiple-sentences.tex using Listing 330</code></td>
<td>93</td>
</tr>
<tr>
<td>330</td>
<td><code>manipulate-sentences.yaml</code></td>
<td>93</td>
</tr>
<tr>
<td>331</td>
<td><code>multiple-sentences.tex using Listing 332</code></td>
<td>93</td>
</tr>
<tr>
<td>332</td>
<td><code>keep-sen-line-breaks.yaml</code></td>
<td>93</td>
</tr>
<tr>
<td>333</td>
<td><code>sentencesFollow</code></td>
<td>93</td>
</tr>
<tr>
<td>334</td>
<td><code>sentencesBeginWith</code></td>
<td>93</td>
</tr>
<tr>
<td>335</td>
<td><code>sentencesEndWith</code></td>
<td>93</td>
</tr>
<tr>
<td>336</td>
<td><code>multiple-sentences.tex using Listing 337</code></td>
<td>94</td>
</tr>
<tr>
<td>337</td>
<td><code>sentences-follow1.yaml</code></td>
<td>94</td>
</tr>
<tr>
<td>338</td>
<td><code>multiple-sentences1.text</code></td>
<td>94</td>
</tr>
<tr>
<td>339</td>
<td><code>multiple-sentences1.txt using Listing 330 on page 93</code></td>
<td>94</td>
</tr>
<tr>
<td>340</td>
<td><code>multiple-sentences1.text using Listing 341</code></td>
<td>94</td>
</tr>
<tr>
<td>341</td>
<td><code>sentences-follow2.yaml</code></td>
<td>94</td>
</tr>
<tr>
<td>342</td>
<td><code>multiple-sentences2.text</code></td>
<td>95</td>
</tr>
<tr>
<td>343</td>
<td><code>multiple-sentences2.text using Listing 330 on page 93</code></td>
<td>95</td>
</tr>
<tr>
<td>344</td>
<td><code>multiple-sentences2.tex using Listing 345</code></td>
<td>95</td>
</tr>
<tr>
<td>345</td>
<td><code>sentences-begin1.yaml</code></td>
<td>95</td>
</tr>
<tr>
<td>346</td>
<td><code>multiple-sentences.tex using Listing 347</code></td>
<td>95</td>
</tr>
<tr>
<td>347</td>
<td><code>sentences-end1.yaml</code></td>
<td>95</td>
</tr>
<tr>
<td>348</td>
<td><code>multiple-sentences.tex using Listing 349</code></td>
<td>96</td>
</tr>
<tr>
<td>349</td>
<td><code>sentences-end2.yaml</code></td>
<td>96</td>
</tr>
<tr>
<td>350</td>
<td><code>url.txt</code></td>
<td>96</td>
</tr>
<tr>
<td>351</td>
<td><code>url.txt using Listing 330 on page 93</code></td>
<td>96</td>
</tr>
<tr>
<td>352</td>
<td><code>url.txt using Listing 353</code></td>
<td>97</td>
</tr>
<tr>
<td>353</td>
<td><code>alt-full-stop1.yaml</code></td>
<td>97</td>
</tr>
<tr>
<td>354</td>
<td><code>multiple-sentences3.txt</code></td>
<td>97</td>
</tr>
<tr>
<td>355</td>
<td><code>multiple-sentences3.tex using Listing 330 on page 93</code></td>
<td>97</td>
</tr>
<tr>
<td>356</td>
<td><code>multiple-sentences4.txt</code></td>
<td>98</td>
</tr>
<tr>
<td>357</td>
<td><code>multiple-sentences4.txt using Listing 330 on page 93</code></td>
<td>98</td>
</tr>
<tr>
<td>358</td>
<td><code>multiple-sentences4.txt using Listing 332 on page 93</code></td>
<td>98</td>
</tr>
<tr>
<td>359</td>
<td><code>multiple-sentences4.txt using Listing 360</code></td>
<td>98</td>
</tr>
<tr>
<td>360</td>
<td><code>item-rules2.yaml</code></td>
<td>98</td>
</tr>
<tr>
<td>361</td>
<td><code>item-rules5.yaml</code></td>
<td>98</td>
</tr>
<tr>
<td>362</td>
<td><code>multiple-sentences5.txt</code></td>
<td>99</td>
</tr>
<tr>
<td>363</td>
<td><code>multiple-sentences5.txt using Listing 363</code></td>
<td>99</td>
</tr>
<tr>
<td>364</td>
<td><code>multiple-sentences6.txt</code></td>
<td>99</td>
</tr>
<tr>
<td>365</td>
<td><code>multiple-sentences6-mod1.txt using Listing 363</code></td>
<td>99</td>
</tr>
<tr>
<td>366</td>
<td><code>multiple-sentences6-mod2.txt using Listing 363 and no sentence indentation</code></td>
<td>99</td>
</tr>
<tr>
<td>367</td>
<td><code>itemize.yaml</code></td>
<td>100</td>
</tr>
<tr>
<td>368</td>
<td><code>multiple-sentences6-mod3.txt using Listing 363 and Listing 367</code></td>
<td>100</td>
</tr>
<tr>
<td>369</td>
<td><code>environments</code></td>
<td>101</td>
</tr>
<tr>
<td>370</td>
<td><code>env-mlb1.txt</code></td>
<td>101</td>
</tr>
<tr>
<td>371</td>
<td><code>env-mlb1.yaml</code></td>
<td>101</td>
</tr>
<tr>
<td>372</td>
<td><code>env-mlb2.yaml</code></td>
<td>101</td>
</tr>
<tr>
<td>373</td>
<td><code>env-mlb2.txt using Listing 371</code></td>
<td>101</td>
</tr>
<tr>
<td>374</td>
<td><code>env-mlb2.txt using Listing 372</code></td>
<td>101</td>
</tr>
<tr>
<td>375</td>
<td><code>env-mlb3.yaml</code></td>
<td>101</td>
</tr>
<tr>
<td>376</td>
<td><code>env-mlb4.yaml</code></td>
<td>101</td>
</tr>
<tr>
<td>377</td>
<td><code>env-mlb4.txt using Listing 375</code></td>
<td>101</td>
</tr>
<tr>
<td>378</td>
<td><code>env-mlb4.txt using Listing 376</code></td>
<td>101</td>
</tr>
<tr>
<td>379</td>
<td><code>env-mlb5.yaml</code></td>
<td>102</td>
</tr>
<tr>
<td>380</td>
<td><code>env-mlb5.yaml</code></td>
<td>102</td>
</tr>
<tr>
<td>381</td>
<td><code>env-mlb5.txt using Listing 379</code></td>
<td>102</td>
</tr>
<tr>
<td>382</td>
<td><code>env-mlb5.txt using Listing 380</code></td>
<td>102</td>
</tr>
</tbody>
</table>
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 \LaTeX\ stack exchange [1] I received some positive feedback and follow-up feature requests. A big thank you to Harish Kumar [15] 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 11.2 on page 138 for their valued contributions, and thank you to those who report bugs and request features at [11].

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 26) 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 10). 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{quote}
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{quote}

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

\begin{warning}
Before you try the script on anything important (like your thesis), test it out on the sample files in the \texttt{test-case} directory [11].
\end{warning}

If you have used any version 2.\* of \texttt{latexindent.pl}, there are a few changes to the interface; see appendix F on page 148 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 539. 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{listing}
\begin{verbatim}
\texttt{demo-tex.tex}
\end{verbatim}
\end{listing}

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

```
~/ $ 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.
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.

```bash
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, for example, as well as appendix A on page 140.

### 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 [10].
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 [27]

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

<table>
<thead>
<tr>
<th>LISTING 8: tikzset.tex</th>
<th>LISTING 9: tikzset.tex default output</th>
</tr>
</thead>
<tbody>
<tr>
<td>\tikzset{</td>
<td>\tikzset{</td>
</tr>
<tr>
<td>shrink inner sep/.code={</td>
<td>shrink inner sep/.code={</td>
</tr>
<tr>
<td>\pgfkeysgetvalue...</td>
<td>\pgfkeysgetvalue...</td>
</tr>
<tr>
<td>\pgfkeysgetvalue...</td>
<td>\pgfkeysgetvalue...</td>
</tr>
<tr>
<td>}</td>
<td>}</td>
</tr>
<tr>
<td>}</td>
<td>}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LISTING 10: pstricks.tex</th>
<th>LISTING 11: pstricks.tex default output</th>
</tr>
</thead>
<tbody>
<tr>
<td>\def\Picture#1{%</td>
<td>\def\Picture#1{%</td>
</tr>
<tr>
<td>\def\stripH[#1]{%</td>
<td>\def\stripH[#1]{%</td>
</tr>
<tr>
<td>\begin{pspicture}</td>
<td>\begin{pspicture}</td>
</tr>
<tr>
<td>[showgrid]</td>
<td>[showgrid]</td>
</tr>
<tr>
<td>\psforeach{\row}{%</td>
<td>\psforeach{\row}{%</td>
</tr>
<tr>
<td>{{3,2.8,2.7,3,3.1}},%</td>
<td>{{3,2.8,2.7,3,3.1}},%</td>
</tr>
<tr>
<td>{2.8,1,1.2,2,3},%</td>
<td>{2.8,1,1.2,2,3},%</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>}%</td>
<td>}%</td>
</tr>
<tr>
<td>\expandafter...</td>
<td>\expandafter...</td>
</tr>
<tr>
<td>}</td>
<td>}</td>
</tr>
<tr>
<td>\end{pspicture}}</td>
<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 [11] should you wish to use them without a TeX distribution; in this case, you may like to read appendix B on page 143 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 25.

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 140 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
$ latexindent.pl -v
```

This will output only the version number to the terminal.

-h, --help

```bash
$ latexindent.pl -h
```

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

```bash
$ 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

```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 cannot 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=output.tex, --outputfile=output.tex

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

Note that using -o as above is equivalent to using

```
$ latexindent.pl myfile.tex > output.tex
```

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:

```
$ 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:

```
$ 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,

```
$ 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

```
$ latexindent.pl myfile.tex -o=+
```

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

```
$ latexindent.pl myfile.tex -o=++
```

¹Users of version 2.* should note the subtle change in syntax
tells \texttt{latexindent.pl} to output to \texttt{myfileout0.tex}, but if it exists, then try \texttt{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 \textit{after} the ++ symbols, for example

\begin{verbatim}
$ latexindent.pl myfiletex -o=++out++.tex
\end{verbatim}

See appendix F on page 148 for details of how the interface has changed from Version 2.2 to Version 3.0 for this flag.

\texttt{-s, --silent}

\begin{verbatim}
$ latexindent.pl -s myfile.tex
$ latexindent.pl myfile.tex -s
\end{verbatim}

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

\texttt{-t, --trace}

\begin{verbatim}
$ latexindent.pl -t myfile.tex
$ latexindent.pl myfile.tex -t
\end{verbatim}

Tracing mode: verbose output will be given to \texttt{indent.log}. This is useful if \texttt{latexindent.pl} has made a mistake and you're trying to find out where and why. You might also be interested in learning about \texttt{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.

\texttt{-tt, --ttrace}

\begin{verbatim}
$ latexindent.pl -tt myfile.tex
$ latexindent.pl myfile.tex -tt
\end{verbatim}

More detailed tracing mode: this option gives more details to \texttt{indent.log} than the standard trace option (note that, even more so than with \texttt{-t}, especially for large files, performance of the script will be affected).

\texttt{-l, --local[=myyaml.yaml,other.yaml,...]}

\begin{verbatim}
$ latexindent.pl \texttt{-l} myfile.tex
$ latexindent.pl \texttt{-l=myyaml.yaml} myfile.tex
$ latexindent.pl \texttt{-l first.yaml,second.yaml,third.yaml} myfile.tex
$ latexindent.pl \texttt{-l=first.yaml,second.yaml,third.yaml} myfile.tex
\end{verbatim}

\texttt{latexindent.pl} will always load \texttt{defaultSettings.yaml} (rhymes with camel) and if it is called with the \texttt{-l} switch and it finds \texttt{localSettings.yaml} in the same directory as \texttt{myfile.tex}, then, if not found, it looks for \texttt{localSettings.yaml} and friends, see Section 4.2 on page 22) in the current working directory, then these settings will be added to the indentation scheme. Information will be given in \texttt{indent.log} on the success or failure of loading \texttt{localSettings.yaml}.

The \texttt{-l} flag can take an \texttt{optional} parameter which details the name (or names separated by commas) of a YAML file(s) that resides in the same directory as \texttt{myfile.tex}; you can use this option if you would like to load a settings file in the current working directory that is \textit{not} called \texttt{localSettings.yaml}. In fact, you can specify both \textit{relative} and \textit{absolute paths} for your YAML files; for example
You will find a lot of other explicit demonstrations of how to use the \texttt{-l} switch throughout this documentation.

You can call the \texttt{-l} switch with a \texttt{‘+’} symbol either before or after another YAML file; for example:

```bash
$ latexindent.pl -l=+myyaml.yaml myfile.tex
$ latexindent.pl -l "+myyaml.yaml" myfile.tex
$ latexindent.pl -l=myyaml.yaml+ myfile.tex
```

which translate, respectively, to

```bash
$ latexindent.pl -l=localSettings.yaml,myyaml.yaml myfile.tex
$ latexindent.pl -l=localSettings.yaml,myyaml.yaml myfile.tex
$ latexindent.pl -l=myyaml.yaml,localSettings.yaml myfile.tex
```

Note that the following is \emph{not} allowed:

```bash
$ latexindent.pl -l+myyaml.yaml myfile.tex
```

and

```bash
$ latexindent.pl -l + myyaml.yaml myfile.tex
```

will \emph{only} load \texttt{localSettings.yaml}, and \texttt{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 \texttt{yaml} extension, such as

```bash
$ latexindent.pl -l=localSettings,myyaml myfile.tex
```

\textbf{-y, \texttt{--yaml=yaml settings}}

You can specify YAML settings from the command line using the \texttt{-y} or \texttt{--yaml} switch; sample demonstrations are given above. Note, in particular, that multiple settings can be specified by separating them via commas. There is a further option to use a \texttt{;} to separate fields, which is demonstrated in Section 4.3 on page 23.

Any settings specified via this switch will be loaded after any specified using the \texttt{-l} switch. This is discussed further in Section 4.4 on page 23.

\textbf{-d, \texttt{--onlydefault}}
3.1 From the command line

cmh:~$ latexindent.pl -d myfile.tex

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>

cmh:~$ 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. 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.

If latexindent.pl can not open the log file that you specify, then the script will operate, and no log file will be produced; this might be helpful to users who wish to specify the following, for example

cmh:~$ latexindent.pl -g /dev/null myfile.tex

-s1, --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

cmh:~$ latexindent.pl -m myfile.tex

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 71 latexindent.pl can also be called on a file without the file extension, for example

cmh:~$ latexindent.pl myfile

and in which case, you can specify the order in which extensions are searched for; see Listing 16 on page 25 for full details.

STDIN
3.1 From the command line

```
cmh:~$ cat myfile.tex | latexindent.pl
```
```
cmh:~$ 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:

```
cmh:~$ 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.

```
cmh:~$ 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 [4] for an update to this feature.

- r, -replacement

```
cmh:~$ latexindent.pl -r myfile.tex
```
```
cmh:~$ 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 117.

- rv, -replacementrespectverb

```
cmh:~$ latexindent.pl -rv myfile.tex
```
```
cmh:~$ 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 117.

- rr, -onlyreplacement

```
cmh:~$ latexindent.pl -rr myfile.tex
```
```
cmh:~$ 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 117.

- k, -check

```
cmh:~$ latexindent.pl -k myfile.tex
```
```
cmh:~$ latexindent.pl -check myfile.tex
```

You can instruct latexindent.pl to check if the text after indentation matches that given in the original file.
The exit code of `latexindent.pl` is 0 by default. If you use the `-k` switch then

- if the text after indentation matches that given in the original file, then the exit code is 0;
- if the text after indentation does *not* match that given in the original file, then the exit code is 1.

The value of the exit code may be important to those wishing to, for example, check the status of the indentation in continuous integration tools such as GitHub Actions. Full details of the exit codes of `latexindent.pl` are given in Table 1.

A simple `diff` will be given in `indent.log`.

```bash
$ latexindent.pl -kv myfile.tex
$ latexindent.pl --checkv myfile.tex
```

The check `verbose` switch is exactly the same as the `-k` switch, except that it is `verbose`, and it will output the (simple) `diff` to the terminal, as well as to `indent.log`.

```bash
$ latexindent.pl -n 5-8 myfile.tex
$ latexindent.pl --lines 5-8 myfile.tex
```

The `lines` switch instructs `latexindent.pl` to operate only on specific line ranges within `myfile.tex`.

Complete demonstrations are given in Section 8.

### 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 [3].

### 3.3 Summary of exit codes

Assuming that you call `latexindent.pl` on `myfile.tex`

```bash
$ latexindent.pl myfile.tex
```

then `latexindent.pl` can exit with the exit codes given in Table 1.

<table>
<thead>
<tr>
<th>exit code</th>
<th>indentation</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>✔</td>
<td>success; if <code>-k</code> or <code>-kv</code> active, indented text matches original</td>
</tr>
<tr>
<td>0</td>
<td>✗</td>
<td>success; if <code>-version</code> or <code>-help</code>, no indentation performed</td>
</tr>
<tr>
<td>1</td>
<td>✔</td>
<td>success, and <code>-k</code> or <code>-kv</code> active; indented text different from original</td>
</tr>
<tr>
<td>2</td>
<td>✗</td>
<td>failure, <code>defaultSettings.yaml</code> could not be read</td>
</tr>
<tr>
<td>3</td>
<td>✗</td>
<td>failure, <code>myfile.tex</code> not found</td>
</tr>
<tr>
<td>4</td>
<td>✗</td>
<td>failure, <code>myfile.tex</code> exists but cannot be read</td>
</tr>
<tr>
<td>5</td>
<td>✗</td>
<td>failure, <code>-w</code> active, and back-up file cannot be written</td>
</tr>
<tr>
<td>6</td>
<td>✗</td>
<td>failure, <code>-c</code> active, and cruft directory does not exist</td>
</tr>
</tbody>
</table>
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 [9] 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\usernameListing 12 shows a sample indentconfig.yaml file.

<table>
<thead>
<tr>
<th>Listing 12: indentconfig.yaml (sample)</th>
</tr>
</thead>
<tbody>
<tr>
<td># Paths to user settings for latexindent.pl</td>
</tr>
<tr>
<td>#</td>
</tr>
<tr>
<td># Note that the settings will be read in the order you</td>
</tr>
<tr>
<td># specify here- each successive settings file will overwrite</td>
</tr>
<tr>
<td># the variables that you specify</td>
</tr>
</tbody>
</table>

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 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.

If you find that `latexindent.pl` does not read your `.yaml` file, then it might be as a result of the default commandline encoding not being UTF-8; normally this will only occur for Windows users. In this case, you might like to explore the encoding option for `indentconfig.yaml` as demonstrated in Listing 14.

```yaml
encoding: GB2312
paths:
  - D:\cmh\latexindent.yaml
```

Thank you to [22] for this contribution; please see appendix D on page 146 and details within [21] for further information.

### 4.2 localSettings.yaml and friends

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

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

then `latexindent.pl` will search for and then, assuming they exist, load each of the following files in the following order:

1. `localSettings.yaml`
2. `latexindent.yaml`
3. `.localSettings.yaml`
4. `.latexindent.yaml`

These files will be assumed to be in the same directory as `myfile.tex`, or otherwise in the current working directory. You do not need to have all of the above files, usually just one will be sufficient. In what follows, whenever we refer to `localSettings.yaml` it is assumed that it can mean any of the four named options listed above.

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,

3 Windows users may find that they have to end `.yaml` files with a blank line.
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 15 using the -y switch, then you could use the following command:

```bash
cmh:~$ latexindent.pl -y="verbatimEnvironments:cmhenv:0;myenv:1" myfile.tex
```

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

```bash
cmh:~$ latexindent.pl
   -y="verbatimEnvironments:cmhenv:0,verbatimEnvironments: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,

```bash
cmh:~$ latexindent.pl -l=mysettings.yaml
   -y="verbatimEnvironments:cmhenv:0;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 6.5 on page 91) and the listings within Listing 335 on page 93, the following settings give the option to have sentences end with a semicolon

```bash
cmh:~$ latexindent.pl -m
   --yaml="modifyLineBreaks:oneSentencePerLine:sentencesEndWith:other:;'
```

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.

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 15, and you’ll find plenty of further examples throughout this manual.

---

**Listing 15: localSettings.yaml (example)**

```yaml
# verbatim environments - environments specified
# here will not be changed at all!
verbatimEnvironments:
  cmhenv: 0
  myenv: 1
```

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.
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.

---

**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.
latexindent.pl loads its settings from defaultSettings.yaml. The idea is to separate the behavior 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 behavior 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.

For most of the settings in defaultSettings.yaml that are specified as integers, then we understand 0 to represent 'off' and 1 to represent 'on'. For fields that allow values other than 0 or 1, it is hoped that the specific context and associated commentary should make it clear which values are allowed.

### List 16: fileExtensionPreference

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>44</td>
<td>fileExtensionPreference:</td>
</tr>
<tr>
<td>45</td>
<td>.tex: 1</td>
</tr>
<tr>
<td>46</td>
<td>.sty: 2</td>
</tr>
<tr>
<td>47</td>
<td>.cls: 3</td>
</tr>
<tr>
<td>48</td>
<td>.bib: 4</td>
</tr>
</tbody>
</table>

Calling latexindent.pl myfile with the (default) settings specified in Listing 16 means that the script will first look for myfile.tex, then myfile.sty, myfile.cls, and finally myfile.bib in order.

### 5.1 Backup and log file preferences

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.

---

4 Throughout this manual, listings shown with line numbers represent code taken directly from defaultSettings.yaml.
5.1 Backup and log file preferences

```plaintext
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.

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.

```cmh
$ copy myfile.bak1 to myfile.bak0
$cmb:$ copy myfile.bak2 to myfile.bak1
$cmb:$ copy myfile.bak3 to myfile.bak2
$cmb:$ copy myfile.bak4 to myfile.bak3
```

The default value of cycleThroughBackUps is 0.

logFilePreferences: (fields)

latexindent.pl writes information to indent.log, some of which can be customized by changing logFilePreferences; see Listing 17. If you load your own user settings (see Section 4 on page 21) 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 in the log file by switching showAmalgamatedSettings to 1, if you wish.

```
<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>88</td>
<td>logFilePreferences:</td>
</tr>
<tr>
<td>89</td>
<td>showEveryYamlRead: 1</td>
</tr>
<tr>
<td>90</td>
<td>showAmalgamatedSettings: 0</td>
</tr>
<tr>
<td>91</td>
<td>showDecorationStartCodeBlockTrace: 0</td>
</tr>
<tr>
<td>92</td>
<td>showDecorationFinishCodeBlockTrace: 0</td>
</tr>
<tr>
<td>93</td>
<td>endLogFileWith: '--------------'</td>
</tr>
<tr>
<td>94</td>
<td>showGitHubInfoFooter: 1</td>
</tr>
<tr>
<td>95</td>
<td>Dumper:</td>
</tr>
<tr>
<td>96</td>
<td>Terse: 1</td>
</tr>
<tr>
<td>97</td>
<td>Indent: 1</td>
</tr>
<tr>
<td>98</td>
<td>Useqq: 1</td>
</tr>
<tr>
<td>99</td>
<td>Deparse: 1</td>
</tr>
<tr>
<td>100</td>
<td>Quotekeys: 0</td>
</tr>
<tr>
<td>101</td>
<td>Sortkeys: 1</td>
</tr>
<tr>
<td>102</td>
<td>Pair: &quot; =&gt; &quot;</td>
</tr>
</tbody>
</table>
```

When either of the trace modes (see page 16) 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 145.
5.2 Verbatim code blocks

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.

Note: latexindent.pl no longer uses the log4perl module to handle the creation of the logfile.

Some of the options for Perl's Dumper module can be specified in Listing 17; see [7] and [6] for more information. These options will mostly be helpful for those calling latexindent.pl with the -tt option described in Section 3.1.

5.2 Verbatim code blocks

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 18.

<table>
<thead>
<tr>
<th>Listing 18: verbatimEnvironments</th>
<th>Listing 19: verbatimCommands</th>
</tr>
</thead>
<tbody>
<tr>
<td>106 verbatimEnvironments:</td>
<td>112 verbatimCommands:</td>
</tr>
<tr>
<td>107     verbatim: 1</td>
<td>113     verb: 1</td>
</tr>
<tr>
<td>108     lstlisting: 1</td>
<td>114     lstinline: 1</td>
</tr>
<tr>
<td>109     minted: 1</td>
<td></td>
</tr>
</tbody>
</table>

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.

verbatimCommands: (fields)

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 71).

With reference to Listing 19, by default latexindent.pl looks for \verb immediately followed by another character, and then it takes the body as anything up to the next occurrence of the character; this means that, for example, \verb!x+3! is treated as a verbatimCommands.

noIndentBlock: (fields)

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 20.

<table>
<thead>
<tr>
<th>Listing 20: noIndentBlock</th>
</tr>
</thead>
<tbody>
<tr>
<td>119 noIndentBlock:</td>
</tr>
<tr>
<td>120     noindent: 1</td>
</tr>
<tr>
<td>121     cmhtest: 1</td>
</tr>
</tbody>
</table>

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 21 for example.
### Listing 21: noIndentBlock.tex

```latex
\begin{noindent}
some before text
  this code
  won't
  be touched
  
  by
  \texttt{latexindent.pl}!
\end{noindent}

some after text
\end{noindent}
```

Important note: it is assumed that the noindent block statements specified in this way appear on their own line.

The `noIndentBlock` fields can also be specified in terms of `begin` and `end` fields. We use the code in Listing 22 to demonstrate this feature.

### Listing 22: noIndentBlock1.tex

```latex
some before text
  this code
  won't
  be touched
  
  by
  \texttt{latexindent.pl}!

some after text
```

The settings given in Listings 23 and 24 are equivalent:

<table>
<thead>
<tr>
<th>Listing 23: noindent1.yaml</th>
<th>Listing 24: noindent2.yaml</th>
<th>Listing 25: noindent3.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>noIndentBlock:</td>
<td>noIndentBlock:</td>
<td>noIndentBlock:</td>
</tr>
<tr>
<td>demo:</td>
<td>demo:</td>
<td>demo:</td>
</tr>
<tr>
<td>begin: 'some\hbefore'</td>
<td>begin: 'some\hbefore'</td>
<td>begin: 'some\hbefore'</td>
</tr>
<tr>
<td>body: '.*?'</td>
<td>end: 'some\hafter\htext'</td>
<td></td>
</tr>
<tr>
<td>end: 'some\hafter\htext'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>lookForThis: 1</td>
<td></td>
<td>lookForThis: 0</td>
</tr>
</tbody>
</table>

Upon running the commands

```
cmh:~$ latexindent.pl -l noindent1.yaml noindent1
cmh:~$ latexindent.pl -l noindent2.yaml noindent1
```

then we receive the output given in Listing 26.

### Listing 26: noIndentBlock1.tex using Listing 23 or Listing 24

```latex
some before text
  this code
  won't
  be touched
  
  by
  \texttt{latexindent.pl}!

some after text
```

The `begin`, `body` and `end` fields for `noIndentBlock` are all regular expressions. If the `body` field is not specified, then it takes a default value of `.*?` which is written explicitly in Listing 23. In this context, we interpret `.*?` in words as the fewest number of characters (possibly none) until the `end` field is reached.

The `lookForThis` field is optional, and can take the values 0 (off) or 1 (on); by default, it is assumed to be 1 (on).

Using Listing 25 demonstrates setting `lookForThis` to 0 (off); running the command

```
cmh:~$ latexindent.pl -l noindent3.yaml noindent1
```
5.3 filecontents and preamble

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 fileContentsEnvironments, see Listing 28. The behaviour of latexindent.pl on these environments is determined by their location (preamble or not), and the value indentPreamble, discussed next.

Listing 28: fileContentsEnvironments

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>125</td>
<td>fileContentsEnvironments:</td>
</tr>
<tr>
<td>126</td>
<td>filecontents: 1</td>
</tr>
<tr>
<td>127</td>
<td>filecontents*: 1</td>
</tr>
</tbody>
</table>

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 indentPreamble is set to 0, by default), but if you'd like latexindent.pl to try then change indentPreamble to 1.

lookForPreamble: (fields)

Not all files contain preamble; for example, sty, cls and bib files typically do not. Referencing Listing 29, if you set, for example, .tex to 0, then regardless of the setting of the value of indentPreamble, preamble will not be assumed when operating upon .tex files.

Listing 29: lookForPreamble

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>133</td>
<td>lookForPreamble:</td>
</tr>
<tr>
<td>134</td>
<td>.tex: 1</td>
</tr>
<tr>
<td>135</td>
<td>.sty: 0</td>
</tr>
<tr>
<td>136</td>
<td>.cls: 0</td>
</tr>
<tr>
<td>137</td>
<td>.bib: 0</td>
</tr>
</tbody>
</table>

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 30.
5.4 Indentation and horizontal space

5.4 Indentation and horizontal space

**defaultIndent:** *(horizontal space)*

This is the default indentation used in the absence of other details for the code block with which we are working. The default value is `\t` which means a tab; we will explore customisation beyond `defaultIndent` in Section 5.8 on page 48.

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

**removeTrailingWhitespace:** *(fields)*

Trailing white space can be removed both before and after processing the document, as detailed in Listing 31; each of the fields can take the values 0 or 1. See Listings 418 to 420 on page 106 for before and after results. Thanks to [28] for providing this feature.

**Listing 31:** removeTrailingWhitespace

<table>
<thead>
<tr>
<th>150</th>
<th>151</th>
<th>152</th>
</tr>
</thead>
<tbody>
<tr>
<td>removeTrailingWhitespace:</td>
<td>beforeProcessing: 0</td>
<td>afterProcessing: 1</td>
</tr>
</tbody>
</table>

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 32.

5.5 Aligning at delimiters

**lookForAlignDelims:** *(fields)*

This contains a list of code blocks that are operated upon in a special way by `latexindent.pl` (see Listing 33). In fact, the fields in `lookForAlignDelims` can actually take two different forms: the basic version is shown in Listing 33 and the advanced version in Listing 36; we will discuss each in turn.

**Listing 33:** lookForAlignDelims (basic)

```latex
lookForAlignDelims:
  tabular: 1
  tabularx: 1
  longtable: 1
  array: 1
  matrix: 1
  ...
```

Specifying code blocks in this field instructs `latexindent.pl` to try and align each column by its alignment delimiters. It does have some limitations (discussed further in Section 10), but in many cases it will produce results such as those in Listings 34 and 35.
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 20 on page 27).

<table>
<thead>
<tr>
<th>Listing 34: tabular1.tex</th>
<th>Listing 35: tabular1.tex default output</th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{tabular}{cccc} 1 &amp; 2 &amp; 3 &amp; 4 \ 5 &amp; 6 &amp; \ \end{tabular}</td>
<td>\begin{tabular}{cccc} 1 &amp; 2 &amp; 3 &amp; 4 \ 5 &amp; 6 &amp; \end{tabular}</td>
</tr>
</tbody>
</table>

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 36 is for you.

<table>
<thead>
<tr>
<th>Listing 36: lookForAlignDelims (advanced)</th>
</tr>
</thead>
<tbody>
<tr>
<td>155 lookForAlignDelims:</td>
</tr>
<tr>
<td>156 tabular:</td>
</tr>
<tr>
<td>157   delims: 1</td>
</tr>
<tr>
<td>158   alignDoubleBackSlash: 1</td>
</tr>
<tr>
<td>159   spacesBeforeDoubleBackSlash: 1</td>
</tr>
<tr>
<td>160   multiColumnGrouping: 0</td>
</tr>
<tr>
<td>161   alignRowsWithoutMaxDelims: 1</td>
</tr>
<tr>
<td>162   spacesBeforeAmpersand: 1</td>
</tr>
<tr>
<td>163   spacesAfterAmpersand: 1</td>
</tr>
<tr>
<td>164   justification: left</td>
</tr>
<tr>
<td>165   alignFinalDoubleBackSlash: 0</td>
</tr>
<tr>
<td>166   dontMeasure: 0</td>
</tr>
<tr>
<td>167   delimiterRegEx: '([^&lt;/]&amp;)'</td>
</tr>
<tr>
<td>168   delimiterJustification: left</td>
</tr>
<tr>
<td>169 tabularx:</td>
</tr>
<tr>
<td>170   delims: 1</td>
</tr>
<tr>
<td>171   longtable: 1</td>
</tr>
</tbody>
</table>

Note that you can use a mixture of the basic and advanced form: in Listing 36 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 33. 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);
- multiColumnGrouping: binary switch (0 or 1) that details if latexindent.pl should group columns above and below a \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 left or 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 5.5.4.

We will explore most of these features using the file tabular2.tex in Listing 37 (which contains a `\multicolumn` command), and the YAML files in Listings 38 to 44; we will explore alignFinalDoubleBackSlash in Listing 65; the dontMeasure feature will be described in Section 5.5.3, and delimiterRegEx in Section 5.5.4.

**LISTING 37: tabular2.tex**

```latex
\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 38: tabular2.yaml**

```yaml
lookForAlignDelims:
  tabular:
    multiColumnGrouping: 1
```

**LISTING 39: tabular3.yaml**

```yaml
lookForAlignDelims:
  tabular:
    alignRowsWithoutMaxDelims: 0
```

**LISTING 40: tabular4.yaml**

```yaml
lookForAlignDelims:
  tabular:
    spacesBeforeAmpersand: 4
```

**LISTING 41: tabular5.yaml**

```yaml
lookForAlignDelims:
  tabular:
    spacesAfterAmpersand: 4
```

**LISTING 42: tabular6.yaml**

```yaml
lookForAlignDelims:
  tabular:
    alignDoubleBackSlash: 0
```

**LISTING 43: tabular7.yaml**

```yaml
lookForAlignDelims:
  tabular:
    spacesBeforeDoubleBackSlash: 0
```

**LISTING 44: tabular8.yaml**

```yaml
lookForAlignDelims:
  tabular:
    justification: "right"
```

On running the commands

```
cmh:~$ latexindent.pl tabular2.tex
cy:~$ latexindent.pl tabular2.tex -l tabular2.yaml
cy:~$ latexindent.pl tabular2.tex -l tabular3.yaml
cy:~$ latexindent.pl tabular2.tex -l tabular2.yaml,tabular4.yaml
cy:~$ latexindent.pl tabular2.tex -l tabular2.yaml,tabular5.yaml
cy:~$ latexindent.pl tabular2.tex -l tabular2.yaml,tabular6.yaml
cy:~$ latexindent.pl tabular2.tex -l tabular2.yaml,tabular7.yaml
cy:~$ latexindent.pl tabular2.tex -l tabular2.yaml,tabular8.yaml
```
we obtain the respective outputs given in Listings 45 to 52.

**LISTING 45: tabular2.tex default output**

```latex
\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 46: tabular2.tex using Listing 38**

```latex
\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 47: tabular2.tex using Listing 39**

```latex
\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 48: tabular2.tex using Listings 38 and 40**

```latex
\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 49: tabular2.tex using Listings 38 and 41**

```latex
\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}
```
5.5 Aligning at delimiters

\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 \ \\
\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 \ \\
\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 45 and 46 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 45 the columns have been aligned at the ampersand;
- in Listing 46 the \multicolumn command has grouped the 2 columns beneath and above it, because multiColumnGrouping is set to 1 in Listing 38;
- in Listing 47 rows 3 and 6 have not been aligned at the ampersand, because alignRowsWithoutMaxDelims has been set to 0 in Listing 39; however, the \ have still been aligned;
- in Listing 48 the columns beneath and above the \multicol commands have been grouped (because multiColumnGrouping is set to 1), and there are at least 4 spaces before each aligned ampersand because spacesBeforeAmpersand is set to 4;
- in Listing 49 the columns beneath and above the \multicol commands have been grouped (because multiColumnGrouping is set to 1), and there are at least 4 spaces after each aligned ampersand because spacesAfterAmpersand is set to 4;
- in Listing 50 the \ have not been aligned, because alignDoubleBackSlash is set to 0, otherwise the output is the same as Listing 46;
- in Listing 51 the \ have been aligned, and because spacesBeforeDoubleBackSlash is set to 0, there are no spaces ahead of them; the output is otherwise the same as Listing 46;
- in Listing 52 the cells have been right-justified; note that cells above and below the \multicol statements have still been group correctly, because of the settings in Listing 38.
5.5 Aligning at delimiters

5.5.1 lookForAlignDelims: spacesBeforeAmpersand

The `spacesBeforeAmpersand` can be specified in a few different ways. The basic form is demonstrated in Listing 40, but we can customise the behaviour further by specifying if we would like this value to change if it encounters a leading blank column; that is, when the first column contains only zero-width entries. We refer to this as the advanced form.

We demonstrate this feature in relation to Listing 53; upon running the following command

```
cmh:~$ latexindent.pl aligned1.tex -o=+-default
```

then we receive the default output given in Listing 54.

<table>
<thead>
<tr>
<th>Listing 53: aligned1.tex</th>
<th>Listing 54: aligned1-default.tex</th>
</tr>
</thead>
</table>
| \begin{aligned}
  & a & b, \ \\
  & c & d.
\end{aligned} | \begin{aligned}
  & a & b, \ \\
  & c & d.
\end{aligned} |

The settings in Listings 55 to 58 are all equivalent; we have used the not-yet discussed noAdditionalIndent field (see Section 5.8 on page 48) which will assist in the demonstration in what follows.

<table>
<thead>
<tr>
<th>Listing 55: sba1.yaml</th>
<th>Listing 56: sba2.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>noAdditionalIndent:</td>
<td>noAdditionalIndent:</td>
</tr>
<tr>
<td>aligned: 1</td>
<td>aligned: 1</td>
</tr>
<tr>
<td>lookForAlignDelims:</td>
<td>lookForAlignDelims:</td>
</tr>
<tr>
<td>aligned: 1</td>
<td>aligned: 1</td>
</tr>
<tr>
<td></td>
<td>spacesBeforeAmpersand: 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 57: sba3.yaml</th>
<th>Listing 58: sba4.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>noAdditionalIndent:</td>
<td>noAdditionalIndent:</td>
</tr>
<tr>
<td>aligned: 1</td>
<td>aligned: 1</td>
</tr>
<tr>
<td>lookForAlignDelims:</td>
<td>lookForAlignDelims:</td>
</tr>
<tr>
<td>aligned:</td>
<td>aligned:</td>
</tr>
<tr>
<td>spacesBeforeAmpersand:</td>
<td>default: 1</td>
</tr>
<tr>
<td>leadingBlankColumn:</td>
<td>1</td>
</tr>
</tbody>
</table>

Upon running the following commands

```
cmh:~$ latexindent.pl aligned1.tex -l sba1.yaml
cmh:~$ latexindent.pl aligned1.tex -l sba2.yaml
cmh:~$ latexindent.pl aligned1.tex -l sba3.yaml
cmh:~$ latexindent.pl aligned1.tex -l sba4.yaml
```

then we receive the (same) output given in Listing 59; we note that there is one space before each ampersand.

<table>
<thead>
<tr>
<th>Listing 59: aligned1-mod1.tex</th>
</tr>
</thead>
</table>
| \begin{aligned}
  & a & b, \ \\
  & c & d.
\end{aligned} |

We note in particular:

- Listing 55 demonstrates the basic form for `lookForAlignDelims`; in this case, the default values are specified as in Listing 36 on page 31;
- Listing 56 demonstrates the advanced form for `lookForAlignDelims` and specified `spacesBeforeAmpersand`. The default value is 1;
5.5 Aligning at delimiters

- Listing 57 demonstrates the new advanced way to specify spacesBeforeAmpersand, and for us to set the default value that sets the number of spaces before ampersands which are not in leading blank columns. The default value is 1.

We note that leadingBlankColumn has not been specified in Listing 57, and it will inherit the value from default;

- Listing 58 demonstrates spaces to be used before ampersands for leading blank columns. We note that default has not been specified, and it will be set to 1 by default.

We can customise the space before the ampersand in the leading blank column of Listing 59 by using either of Listings 60 and 61, which are equivalent.

Upon running

```bash
$ latexindent.pl aligned1.tex -l sba5.yaml
$ latexindent.pl aligned1.tex -l sba6.yaml
```

then we receive the (same) output given in Listing 62. We note that the space before the ampersand in the leading blank column has been set to 0 by Listing 61.

We can demonstrated this feature further using the settings in Listing 64 which give the output in Listing 63.

<table>
<thead>
<tr>
<th>LISTING 60: sba5.yaml</th>
<th>LISTING 61: sba6.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>noAdditionalIndent:</td>
<td>noAdditionalIndent:</td>
</tr>
<tr>
<td>aligned: 1</td>
<td>aligned: 1</td>
</tr>
<tr>
<td>lookForAlignDelims:</td>
<td>lookForAlignDelims:</td>
</tr>
<tr>
<td>aligned:</td>
<td>aligned:</td>
</tr>
<tr>
<td>spacesBeforeAmpersand:</td>
<td>spacesBeforeAmpersand:</td>
</tr>
<tr>
<td>leadingBlankColumn: 0</td>
<td>leadingBlankColumn: 0</td>
</tr>
<tr>
<td>default: 1</td>
<td>default: 1</td>
</tr>
</tbody>
</table>

5.5.2 lookForAlignDelims: alignFinalDoubleBackSlash

We explore the alignFinalDoubleBackSlash feature by using the file in Listing 65. Upon running the following commands

```bash
$ latexindent.pl tabular4.tex -o=+-default
$ latexindent.pl tabular4.tex -o=+-FDBS -y="lookForAlignDelims:tabular:alignFinalDoubleBackSlash:1"
```

then we receive the respective outputs given in Listing 66 and Listing 67.

<table>
<thead>
<tr>
<th>LISTING 65: tabular4.tex</th>
<th>LISTING 66: tabular4-default.tex</th>
<th>LISTING 67: tabular4-FDBS.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{tabular}{lc}</td>
<td>\begin{tabular}{lc}</td>
<td>\begin{tabular}{lc}</td>
</tr>
<tr>
<td>Name &amp; \shortstack{Hi \ Lo} \</td>
<td>Name &amp; \shortstack{Hi \ Lo} \</td>
<td>Name &amp; \shortstack{Hi \ Lo} \</td>
</tr>
<tr>
<td>Foo &amp; Bar</td>
<td>Foo &amp; Bar</td>
<td>Foo &amp; Bar</td>
</tr>
<tr>
<td>\end{tabular}</td>
<td>\end{tabular}</td>
<td>\end{tabular}</td>
</tr>
</tbody>
</table>

We note that in:
5.5 Aligning at delimiters

- Listing 66, by default, the first set of double back slashes in the first row of the tabular environment have been used for alignment;
- Listing 67, the final set of double back slashes in the first row have been used, because we specified alignFinalDoubleBackSlash as 1.

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 42); 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 68 and 69 are achievable by default.

<table>
<thead>
<tr>
<th>Listing 68: matrix1.tex</th>
<th>Listing 69: matrix1.tex default output</th>
</tr>
</thead>
</table>
| \matrix [ \ 
| 1 & 2 & 3 \ 
| 4 & 5 & 6] \ 
| 7 & 8 & 9 \ 
| 10 & 11 & 12 \ 
| } | \matrix [ \ 
| 1 & 2 & 3 \ 
| 4 & 5 & 6] \ 
| 7 & 8 & 9 \ 
| 10 & 11 & 12 \ 
| } |

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 70; the default output is shown in Listing 71. 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 70: align-block.tex</th>
<th>Listing 71: align-block.tex default output</th>
</tr>
</thead>
</table>
| %* \begin{tabular} \ 
| 1 & 2 & 3 & 4 \ 
| 5 & 6 & 7 & 8 \ 
| } | %* \begin{tabular} \ 
| 1 & 2 & 3 & 4 \ 
| 5 & 6 & 7 & 8 \ 
| } |

With reference to Table 2 on page 49 and the, yet undiscussed, fields of noAdditionalIndent and indentRules (see Section 5.8 on page 48), these comment-marked blocks are considered environments.

5.5.3 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 72; the default output is shown in Listing 73.

<table>
<thead>
<tr>
<th>Listing 72: tabular-DM.tex</th>
<th>Listing 73: tabular-DM.tex default output</th>
</tr>
</thead>
</table>
| \begin{tabular}{cccc} \ 
| aaa & bbb & ccc & dd \ 
| 11 & 2 & 33 & 4 \ 
| 5 & 66 & 7 & 8 \ 
| \end{tabular} | \begin{tabular}{cccc} \ 
| aaa & bbb & ccc & dd \ 
| 11 & 2 & 33 & 4 \ 
| 5 & 66 & 7 & 8 \ 
| \end{tabular} |

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 75, we can run the command

```
cmh:~$ latexindent.pl tabular-DM.tex -l=dontMeasure1.yaml
```

and receive the output given in Listing 74.
5.5 Aligning at delimiters

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 77. On running the following commands,

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

we receive the output in Listing 76.

We note that in Listing 77 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 79 and Listing 80. 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 78.

We note that in:

- Listing 79 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 80 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 82; upon running
we receive the output in Listing 81.

\begin{tabular}{cccc}  
| aaaa & bbb & ccc & dd |  
| 11 & 2 & 33 & 4 |  
| 5 & 66 & 7 & 8 |  
\end{tabular}

Finally, the applyTo field can be specified as row, together with a regex expression. For example, for the settings given in Listing 84, upon running

\begin{tabular}{cccc}  
| aaaa & bbb & ccc & dd |  
| 11 & 2 & 33 & 4 |  
| 5 & 66 & 7 & 8 |  
\end{tabular}

we receive the output in Listing 83.

5.5.4 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: an ampersand, as long as it is not immediately preceded by a backslash.

Important: note the 'capturing' parenthesis in the (&) which are necessary; if you intend to customise this field, then be sure to include them appropriately.

Let's say that we wish to align the code at either the \= or \>. We employ the settings given in Listing 88 and run the command

\begin{tabular}{cccc}  
| aaaa & bbb & ccc & dd |  
| 11 & 2 & 33 & 4 |  
| 5 & 66 & 7 & 8 |  
\end{tabular}

Let's say that we wish to align the code at either the \= or \>. We employ the settings given in Listing 88 and run the command

\begin{tabular}{cccc}  
| aaaa & bbb & ccc & dd |  
| 11 & 2 & 33 & 4 |  
| 5 & 66 & 7 & 8 |  
\end{tabular}

Let's say that we wish to align the code at either the \= or \>. We employ the settings given in Listing 88 and run the command

\begin{tabular}{cccc}  
| aaaa & bbb & ccc & dd |  
| 11 & 2 & 33 & 4 |  
| 5 & 66 & 7 & 8 |  
\end{tabular}

We demonstrate how to customise this with respect to the code given in Listing 85; the default output from latexindent.pl is given in Listing 86.
We note that:

- in Listing 87 the code has been aligned, as intended, at both the `=` and `>`;
- in Listing 88 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 90 and run the command

cmh:~$ latexindent.pl tabbing.tex -l=delimiterRegEx2.yaml

to receive the output given in Listing 89.

We note that only the `>` have been aligned.

Of course, the other `lookForAlignDelims` options can be used alongside the `delimiterRegEx`; regardless of the type of delimiter being used (ampersand or anything else), the fields from Listing 36 on page 31 remain the same; for example, using the settings in Listing 92, and running

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

we receive the output in Listing 94.

It is possible that delimiters specified within `delimiterRegEx` can be of different lengths. Consider the file in Listing 93, and associated YAML in Listing 95. Note that the Listing 95 specifies the option for the delimiter to be either `#` or `>`, which are different lengths. Upon running the command

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

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 97 and running the command

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

gives the output in Listing 96.

Note that in Listing 96 the second set of delimiters have been right aligned – it is quite subtle!

### 5.6 Indent after items, specials and headings

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 99 and 100.

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 21 for details of how to configure user settings, and Listing 13 on page 22 in particular.)
5.6 Indent after items, specials and headings

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 102 shows the default settings of `specialBeginEnd`.

<table>
<thead>
<tr>
<th>LISTING 102: specialBeginEnd</th>
</tr>
</thead>
<tbody>
<tr>
<td>244</td>
</tr>
<tr>
<td>245</td>
</tr>
<tr>
<td>246</td>
</tr>
<tr>
<td>247</td>
</tr>
<tr>
<td>248</td>
</tr>
<tr>
<td>249</td>
</tr>
<tr>
<td>250</td>
</tr>
<tr>
<td>251</td>
</tr>
<tr>
<td>252</td>
</tr>
<tr>
<td>253</td>
</tr>
<tr>
<td>254</td>
</tr>
<tr>
<td>255</td>
</tr>
<tr>
<td>256</td>
</tr>
<tr>
<td>257</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 22); 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 103 and 104.

<table>
<thead>
<tr>
<th>LISTING 103: special1.tex before</th>
<th>LISTING 104: special1.tex default output</th>
</tr>
</thead>
<tbody>
<tr>
<td>The function $f$ has formula $f(x)=x^2$.</td>
<td>The function $f$ has formula $f(x)=x^2$.</td>
</tr>
<tr>
<td>If you like splitting dollars, $g(x)=f(2x)$</td>
<td>If you like splitting dollars, $g(x)=f(2x)$</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 105.

<table>
<thead>
<tr>
<th>LISTING 105: specialLR.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{equation} \left[ \sqrt{a+b} \right] \end{equation}</td>
</tr>
</tbody>
</table>

Now consider the YAML files shown in Listings 106 and 107.
5.6 Indent after items, specials and headings

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 108 and 109.

**Listing 108: specialLR.tex using Listing 106**

```
\begin{equation}
\left[\sqrt{a+b}\right]
\end{equation}
```

**Listing 109: specialLR.tex using Listings 106 and 107**

```
\begin{equation}
\left[\sqrt{a+b}\right]
\end{equation}
```

Notice that in:

- Listing 108 the `\left` has been treated as a *command*, with one optional argument;
- Listing 109 the `specialBeginEnd` pattern in Listing 106 has been obeyed because Listing 107 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 110.

**Listing 110: special2.tex**

```
\If
\thing 0
\ElseIf
\thing 1
\ElseIf
\thing 2
\ElseIf
\thing 3
\Else
\thing 4
\EndIf
```

Upon saving the YAML settings in Listings 111 and 113 and running the commands

```
cmh:~$ latexindent.pl special2.tex -l=middle
```
```
cmh:~$ latexindent.pl special2.tex -l=middle1
```
then we obtain the output given in Listings 112 and 114.
5.6 Indent after items, specials and headings

We note that:

- in Listing 112 the bodies of each of the If statements have been indented appropriately;
- the Else statement has not been indented appropriately in Listing 112 – read on!
- we have specified multiple settings for the middle field using the syntax demonstrated in Listing 113 so that the body of the Else statement has been indented appropriately in Listing 114.

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 116 and the YAML in Listing 115, and running

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

then the output in Listing 116 is unchanged.

We can combine the specialBeginEnd with the lookForAlignDelims feature. We begin with the code in Listing 117.

\[
\begin{align*}
\text{special code blocks can be treated as verbatim}
\end{align*}
\]
### Listing 117: `special-align.tex`

```latex
\begin{tikzpicture}
    \path (A) edge node {$0,1,L$} (B)
    edge node {$1,1,R$} (C)
    (B) edge [loop above] node {$1,1,L$} (B)
    edge node {$0,1,L$} (C)
    (C) edge node {$0,1,L$} (D)
    edge [bend left] node {$1,0,R$} (E)
    (D) edge [loop below] node {$1,1,R$} (D)
    edge node {$0,1,R$} (A)
    (E) edge [bend left] node {$1,0,R$} (A);
\end{tikzpicture}
```

Let’s assume that our goal is to align the code at the `edge` and `node` text; we employ the code given in Listing 118 and run the command

```
cmh:~$ latexindent.pl special-align.tex -l edge-node1.yaml -o=+-mod1
```

to receive the output in Listing 119.

#### Listing 118: `edge-node1.yaml`

```yaml
specialBeginEnd: path:
    begin: '\\path'    end: '\
    lookForThis: 1
specialBeforeCommand: 1

lookForAlignDelims: path:
    delimiterRegEx: '(edge|node)'`

The output in Listing 119 is not quite ideal. We can tweak the settings within Listing 118 in order to improve the output; in particular, we employ the code in Listing 120 and run the command

```
cmh:~$ latexindent.pl special-align.tex -l edge-node2.yaml -o=+-mod2
```

to receive the output in Listing 121.

#### Listing 120: `edge-node2.yaml`

```yaml
specialBeginEnd: path:
    begin: '\\path'    end: '\
    specialBeforeCommand: 1

lookForAlignDelims: path:
    delimiterRegEx: '(edge|node)\*{[0-9,A-Z]+}\}'
```

The `lookForThis` field can be considered optional; by default, it is assumed to be 1, which is demonstrated in Listing 120.
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 field.\footnote{5}

```
\begin{Verbatim}
  \begin{verbatim}
  indentAfterHeadings: (fields)
  \end{verbatim}
  \end{Verbatim}
```

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 5.8 on page 48); 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).

For example, assuming that you have the code in Listing 123 saved into headings1.yaml, and that you have the text from Listing 124 saved into headings1.tex.

```
\begin{Verbatim}
  \begin{verbatim}
  \begin{lstlisting}
indentAfterHeadings: subsection:
  indentAfterThisHeading: 1
  level: 1
paragraph:
  indentAfterThisHeading: 1
  level: 2
  \end{lstlisting}
\end{verbatim}
\end{Verbatim}
```

```
\begin{Verbatim}
  \begin{verbatim}
\subsection{subsection title}
subsection text
subsection text
\paragraph{paragraph title}
paragraph text
paragraph text
\paragraph{paragraph title}
paragraph text
paragraph text
  \end{verbatim}
\end{Verbatim}
```

If you run the command

```
\begin{Verbatim}
  \begin{verbatim}
  cmhiln$: latexindent.pl headings1.tex -l=headings1.yaml
  \end{verbatim}
\end{Verbatim}
```

then you should receive the output given in Listing 125.

\footnote{5}{There is a slight difference in interface for this field when comparing Version 2.2 to Version 3.0; see appendix F on page 148 for details.}
5.6 Indent after items, specials and headings

**Listing 125: headings1.tex using Listing 123**

```latex
\subsection{subsection title}
  \subsection text
  \.sync \paragraph{paragraph title}
    \paragraph text
    \paragraph text
    \paragraph text
    \paragraph text

\paragraph{paragraph title}
  \paragraph text
  \paragraph text
  \paragraph text
  \paragraph text
```

Now say that you modify the YAML from Listing 123 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 126; notice that the paragraph and subsection are at the same indentation level.

**Listing 126: headings1.tex second modification**

```latex
\subsection{subsection title}
  \subsection text
  \paragraph{paragraph title}
    \paragraph text
    \paragraph text
    \paragraph text
    \paragraph text

\paragraph{paragraph title}
  \paragraph text
  \paragraph text
  \paragraph text
  \paragraph text
```

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 [25], and is *off* by default.

For example, consider the example shown in Listing 127 together with the default output shown in Listing 128.

**Listing 127: mult-nested.tex**

```latex
\begin{one}
one
  \begin{two}
two
    \begin{three}
      three
      \begin{four}
        four
      \end{four}
    \end{three}
  \end{two}
\end{one}
```

**Listing 128: mult-nested.tex default output**

```latex
\begin{one}
  \begin{two}
    \begin{three}
      \begin{four}
        four
      \end{four}
    \end{three}
    \begin{two}
      two
      \begin{three}
        three
        \begin{four}
          four
        \end{four}
      \end{three}
      \begin{two}
        two
        \begin{three}
          three
          \begin{four}
            four
          \end{four}
        \end{three}
      \end{two}
    \end{three}
  \end{two}
\end{one}
```

Now say that, for example, you have the `max-indentation1.yaml` from Listing 129 and that you run the following command:

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

You should receive the output shown in Listing 130.
Comparing the output in Listings 128 and 130 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 18 on page 27) or noIndentBlock (see Listing 20 on page 27).

5.7 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 2.

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 2 is discussed in Section 9 on page 132.

5.8 noAdditionalIndent and indentRules

latexindent.pl operates on files by looking for code blocks, as detailed in Section 5.7; for each type of code block in Table 2 on the next page (which we will call a (thing) in what follows) it searches YAML fields for information in the following order:

1. noAdditionalIndent for the name of the current (thing);

2. indentRules for the name of the current (thing);

3. noAdditionalIndentGlobal for the type of the current (thing);

4. indentRulesGlobal for the type of the current (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 2 on the following page; for reference, there follows a list of the code blocks covered.

5.8.1 Environments and their arguments ........................................ 50
5.8.2 Environments with items ...................................................... 56
5.8.3 Commands with arguments .................................................. 57
5.8.4 ifelsefi code blocks ......................................................... 59
5.8.5 specialBeginEnd code blocks ............................................. 61
5.8.6 afterHeading code blocks .................................................. 62
5.8.7 The remaining code blocks .................................................. 64
5.8.7.1 keyEqualsValuesBracesBrackets .................................... 64
## Table 2: 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>
</table>
| environments                      | a-zA-Z@\*0-9_\             | \begin\{myenv\}  
body of myenv  
\end\{myenv\}                                                             |
| optionalArguments                 | inherits name from parent (e.g environment name) | [opt arg text]                                                              |
| mandatoryArguments                | inherits name from parent (e.g environment name) | \{mand arg text\}                                                          |
| commands                          | +a-zA-Z@\*0-9_\             | \mycommand\{arguments\}                                                  |
| keyEqualsValuesBracesBrackets     | a-zA-Z@\*0-9_\.h\{\}:-  | my key/.style=(arguments)                                                |
| namedGroupingBracesBrackets       | 0-9\.a-zA-Z@>><            | in\{arguments\}                                                          |
| UnNamedGroupingBracesBrackets     | No name!                    | { or [ or , or \& or )  
or ( or \$ followed by (arguments)         |
| ifElseFi                          | @a-zA-Z but must begin with either      \if of @@if                      |
|                                  | \begin\{enumerate\}  
\item ...  
\end\{enumerate\}                                                  |
| items                             | User specified, see Listings 98 and 101 on page 41 | \chapter\{title\}  
...  
\section\{title\}                                                      |
| specialBeginEnd                   | User specified, see Listing 102 on page 42 | \begin\{filecontents\}  
...  
\end\{filecontents\}                                                  |
5.8.1 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 131.

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

If we do not wish myenv to receive any additional indentation, we have a few choices available to us, as demonstrated in Listings 132 and 133.

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 134; note in particular that the environment myenv has not received any additional indentation, but that the outer environment has still received indentation.

Upon changing the YAML files to those shown in Listings 135 and 136, 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 137.
5.8 noAdditionalIndent and indentRules

Let's now allow myenv to have some optional and mandatory arguments, as in Listing 138.

Upon running

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

we obtain the output shown in Listing 139; 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 132), 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 140 and 141.
5.8 noAdditionalIndent and indentRules

Listing 140: myenv-noAdd5.yaml

```yaml
noAdditionalIndent:
  myenv:
    body: 0
    optionalArguments: 1
    mandatoryArguments: 0
```

Listing 141: myenv-noAdd6.yaml

```yaml
noAdditionalIndent:
  myenv:
    body: 0
    optionalArguments: 0
    mandatoryArguments: 1
```

Upon running

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

we obtain the respective outputs given in Listings 142 and 143. Note that in Listing 142 the text for the optional argument has not received any additional indentation, and that in Listing 143 the mandatory argument has not received any additional indentation; in both cases, the body has not received any additional indentation.

Listing 142: myenv-args.tex using Listing 140

```latex
\begin{outer}
  \begin{myenv}[\%
    \begin{myenv}[\%
      \begin{myenv}[\%
        \begin{myenv}[\%
          \begin{myenv}[\%
            \begin{myenv}[\%
              \begin{myenv}[\%
                \begin{myenv}[\%
                  \begin{myenv}[\%
                    \begin{myenv}[\%
                      \begin{myenv}[\%
                        \begin{myenv}[\%
                          \begin{myenv}[\%
                        \end{myenv}
                      \end{myenv}
                    \end{myenv}
                  \end{myenv}
                \end{myenv}
              \end{myenv}
            \end{myenv}
          \end{myenv}
        \end{myenv}
      \end{myenv}
    \end{myenv}
  \end{myenv}
\end{outer}
```

Listing 143: myenv-args.tex using Listing 141

```latex
\begin{outer}
  \begin{myenv}[\%
    \begin{myenv}[\%
      \begin{myenv}[\%
        \begin{myenv}[\%
          \begin{myenv}[\%
            \begin{myenv}[\%
              \begin{myenv}[\%
                \begin{myenv}[\%
                  \begin{myenv}[\%
                    \begin{myenv}[\%
                      \begin{myenv}[\%
                        \begin{myenv}[\%
                      \end{myenv}
                    \end{myenv}
                  \end{myenv}
                \end{myenv}
              \end{myenv}
            \end{myenv}
          \end{myenv}
        \end{myenv}
    \end{myenv}
  \end{myenv}
\end{outer}
```

indentRules: (fields)

We may also specify indentation rules for environment code blocks using the indentRules field; see, for example, Listings 144 and 145.

Listing 144: myenv-rules1.yaml

```yaml
indentRules:
  myenv: " "
```

Listing 145: myenv-rules2.yaml

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

On applying either of the following commands,

```bash
$ latexindent.pl myenv.tex -l myenv-rules1.yaml
$ latexindent.pl myenv.tex -l myenv-rules2.yaml
```

we obtain the output given in Listing 146; note in particular that the environment myenv has received one tab (from the outer environment) plus three spaces from Listing 144 or 145.
If you specify a field in `indentRules` using anything other than horizontal space, it will be ignored.

Returning to the example in Listing 138 that contains optional and mandatory arguments. Upon using Listing 144 as in

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

we obtain the output in Listing 147; note that the body, optional argument and mandatory argument of `myenv` have all received the same customised indentation.

```
\begin{outer}
  |\begin{myenv}
        \text{body of environment}
        \text{body of environment}
        \text{body of environment}
  \end{myenv}
|\end{outer}
```

You can specify different indentation rules for the different features using, for example, Listings 148 and 149

```
<table>
<thead>
<tr>
<th>Listing 148: myenv-rules3.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>indentRules:</td>
</tr>
<tr>
<td>myenv:</td>
</tr>
<tr>
<td>body: &quot; &quot;</td>
</tr>
<tr>
<td>optionalArguments: &quot; &quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 149: myenv-rules4.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>indentRules:</td>
</tr>
<tr>
<td>myenv:</td>
</tr>
<tr>
<td>body: &quot; &quot;</td>
</tr>
</tbody>
</table>
|    mandatoryArguments: "\t\t"
```

After running

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

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

then we obtain the respective outputs given in Listings 150 and 151.
5.8 noAdditionalIndent and indentRules

Note that in Listing 150, 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 151, 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.

Listing 150: myenv-args.tex using Listing 148
\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}

Listing 151: myenv-args.tex using Listing 149
\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}

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 Listing 152).

Listing 152: noAdditionalIndentGlobal

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 Listing 152).

NoAdditionalIndentGlobal: {fields}

Listing 153: myenv-args.tex using Listing 152
\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}

Listing 154: myenv-args.tex using Listings 144 and 152
\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 155 and 156.
we may run the commands

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

which produces the respective outputs given in Listings 157 and 158. Notice that in Listing 157 the optional argument has not received any additional indentation, and in Listing 158 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 \texttt{latexindent.pl} will make is to look for indentRulesGlobal as detailed in Listing 159.

```
\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}
```

```
indentRulesGlobal: 〈fields
```

If you change the environments field to anything involving horizontal space, say " ", and then run the following commands

```
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 160 and 161. Note that in Listing 160, both the environment blocks have received a single-space indentation, whereas in Listing 161 the outer environment has received single-space indentation (specified by indentRulesGlobal), but myenv has received " ", as specified by the particular indentRules for myenv Listing 144 on page 52.
You can specify `indentRulesGlobal` for both optional and mandatory arguments, as detailed in Listings 162 and 163.

Upon running the following commands:

```
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 164 and 165. Note that the *optional* argument in Listing 164 has received two tabs worth of indentation, while the *mandatory* argument has done so in Listing 165.

5.8.2 Environments with items

With reference to Listings 98 and 101 on page 41, some commands may contain `item` commands; for the purposes of this discussion, we will use the code from Listing 99 on page 41.

Assuming that you’ve populated `itemNames` with the name of your `item`, you can put the item name into `noAdditionalIndent` as in Listing 166, although a more efficient approach may be to change the relevant field in `itemNames` to 0. Similarly, you can customise the indentation that your `item` receives using `indentRules`, as in Listing 167.

```
\begin{outer}
  \begin{myenv}[
    \begin{itemize}
      \item \textbf{optional argument text}
      \item \textbf{optional argument text}
      \item \textbf{mandatory argument text}
      \item \textbf{mandatory argument text}
      \item body of environment
      \item body of environment
    \end{itemize}
  \end{myenv}
\end{outer}
```

```
\begin{outer}
  \begin{myenv}[
    \item \textbf{optional argument text}
    \item \textbf{optional argument text}
    \item \textbf{mandatory argument text}
    \item \textbf{mandatory argument text}
    \item body of environment
    \item body of environment
  \end{myenv}
\end{outer}
```
Upon running the following commands

```bash
cmh:~$ latexindent.pl items1.tex -local item-noAdd1.yaml
cmh:~$ latexindent.pl items1.tex -local item-rules1.yaml
```

the respective outputs are given in Listings 168 and 169; note that in Listing 168 that the text after each item has not received any additional indentation, and in Listing 169, the text after each item has received a single space of indentation, specified by Listing 167.

**Listing 168: items1.tex using Listing 166**

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

**Listing 169: items1.tex using Listing 167**

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

Alternatively, you might like to populate noAdditionalIndentGlobal or indentRulesGlobal using the items key, as demonstrated in Listings 170 and 171. Note that there is a need to 'reset/ remove' the item field from indentRules in both cases (see the hierarchy description given on page 48) as the item command is a member of indentRules by default.

**Listing 170: items-noAdditionalGlobal.yaml**

indentRules:
  item: 0
noAdditionalIndentGlobal:
  items: 1

**Listing 171: items-indentRulesGlobal.yaml**

indentRules:
  item: 0
indentRulesGlobal:
  items: " "

Upon running the following commands,

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

the respective outputs from Listings 168 and 169 are obtained; note, however, that all such item commands without their own individual noAdditionalIndent or indentRules settings would behave as in these listings.

5.8.3 Commands with arguments

Let's begin with the simple example in Listing 172; when latexindent.pl operates on this file, the default output is shown in Listing 173. 

**Listing 172: mycommand.tex**

\mycommand

\{ 
  \mand \arg text 
  \mand \arg text 
\} 

\[ 
  \opt \arg text 
  \opt \arg text 
\]

**Listing 173: mycommand.tex default output**

\mycommand

\{ 
  \mand \arg text 
  \mand \arg text 
\} 

\[ 
  \opt \arg text 
  \opt \arg text 
\]

As in the environment-based case (see Listings 132 and 133 on page 50) we may specify noAdditionalIndent either in 'scalar' form, or in 'field' form, as shown in Listings 174 and 175.

---

6The command code blocks have quite a few subtleties, described in Section 5.9 on page 66.
5.8 noAdditionalIndent and indentRules

After running the following commands,

```
$ latexindent.pl mycommand.tex -l mycommand-noAdd1.yaml
$ latexindent.pl mycommand.tex -l mycommand-noAdd2.yaml
```

we receive the respective output given in Listings 176 and 177.

Note that in Listing 176 that the 'body', optional argument and mandatory argument have all received no additional indentation, while in Listing 177, 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.

We may further customise noAdditionalIndent for mycommand as we did in Listings 140 and 141 on page 52; explicit examples are given in Listings 178 and 179.

After running the following commands,

```
$ latexindent.pl mycommand.tex -l mycommand-noAdd3.yaml
$ latexindent.pl mycommand.tex -l mycommand-noAdd4.yaml
```

we receive the respective output given in Listings 180 and 181.
Attentive readers will note that the body of mycommand in both Listings 180 and 181 has received no additional indent, even though body is explicitly set to 0 in both Listings 178 and 179. This is because, by default, noAdditionalIndentGlobal for commands is set to 1 by default; this can be easily fixed as in Listings 182 and 183.

**LISTING 182:**
```
noAdditionalIndent:
  mycommand:
    body: 0
    optionalArguments: 1
    mandatoryArguments: 0
  noAdditionalIndentGlobal:
    commands: 0
```

**LISTING 183:**
```
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 184 and 185.

**LISTING 184:**
```
\mycommand
  {  
    mand arg text
    mand arg text}
  [  
    opt arg text
    opt arg text  
  ]
```

**LISTING 185:**
```
\mycommand
  {  
    mand arg text
    mand arg text}
  [  
    opt arg text
    opt arg text  
  ]
```

Both indentRules and indentRulesGlobal can be adjusted as they were for environment code blocks, as in Listings 148 and 149 on page 53 and Listings 159, 162 and 163 on pages 55–56.

### 5.8.4 ifelsefi code blocks

Let's use the simple example shown in Listing 186; when latexindent.pl operates on this file, the output as in Listing 187; note that the body of each of the \if statements have been indented, and that the \else statement has been accounted for correctly.

**Listing 186: ifelsefi1.tex**
```
\ifodd\radius
  \ifnum\radius<14
    \pgfmathparse{(100-\radius)*4};
  \else
    \pgfmathparse{(200-\radius)*3};
  \fi
\fi
```

**Listing 187: ifelsefi1.tex default output**
```
\ifodd\radius
  \ifnum\radius<14
    \pgfmathparse{(100-\radius)*4};
  \else
    \pgfmathparse{(200-\radius)*3};
  \fi
\fi
```

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 188 and 189.

**Listing 188: ifnum-noAdd.yaml**
```
noAdditionalIndent:
  ifnum: 1
```

**Listing 189: ifnum-indent-rules.yaml**
```
indentRules:
  ifnum: "    
```
we receive the respective output given in Listings 190 and 191; note that in Listing 190, the ifnum code block has not received any additional indentation, while in Listing 191, the ifnum code block has received one tab and two spaces of indentation.

Listing 190: ifelsefi1.tex using Listing 188

```
\ifodd\radius
  \ifnum\radius<14
    \pgfmathparse{100-(\radius)*4};
  \else
    \pgfmathparse{200-(\radius)*3};
  \fi
\fi
```

Listing 191: ifelsefi1.tex using Listing 189

```
\ifodd\radius
  \ifnum\radius<14
    \pgfmathparse{100-(\radius)*4};
  \else
    \pgfmathparse{200-(\radius)*3};
  \fi
\fi
```

We may specify noAdditionalIndentGlobal and indentRulesGlobal as in Listings 192 and 193.

Listing 192: ifelsefi-noAdd-glob.yaml

```
noAdditionalIndentGlobal:
  ifElseFi: 1
```

Listing 193: ifelsefi-indent-rules-global.yaml

```
indentRulesGlobal:
  ifElseFi: " 
```

Upon 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 outputs in Listings 194 and 195; notice that in Listing 194 neither of the ifelsefi code blocks have received indentation, while in Listing 195 both code blocks have received a single space of indentation.

Listing 194: ifelsefi1.tex using Listing 192

```
\ifodd\radius
  \ifnum\radius<14
    \pgfmathparse{100-(\radius)*4};
  \else
    \pgfmathparse{200-(\radius)*3};
  \fi
\fi
```

Listing 195: ifelsefi1.tex using Listing 193

```
\ifodd\radius
  \ifnum\radius<14
    \pgfmathparse{100-(\radius)*4};
  \else
    \pgfmathparse{200-(\radius)*3};
  \fi
\fi
```

We can further explore the treatment of ifElseFi code blocks in Listing 196, and the associated default output given in Listing 197; note, in particular, that the bodies of each of the 'or statements' have been indented.

Listing 196: ifelsefi2.tex

```
\ifcase#1
  zero\%
  or
  one\%
  or
  two\%
  or
  three\%
  else
  default
\fi
```

Listing 197: ifelsefi2.tex default output

```
\ifcase#1
  zero\%
  or
  one\%
  or
  two\%
  or
  three\%
  else
  default
\fi
```
5.8 noAdditionalIndent and indentRules

5.8.5 specialBeginEnd code blocks

Let’s use the example from Listing 103 on page 42 which has default output shown in Listing 104 on page 42.

It is recommended to specify noAdditionalIndent and indentRules in the ‘scalar’ form for these type of code blocks, although the ‘field’ form would work, assuming that body was specified. Examples are shown in Listings 198 and 199.

After running the following commands,

we receive the respective output given in Listings 200 and 201; note that in Listing 200, the displayMath code block has not received any additional indentation, while in Listing 201, the displayMath code block has received three tabs worth of indentation.

We may specify noAdditionalIndentGlobal and indentRulesGlobal as in Listings 202 and 203.

Upon running the following commands

we receive the outputs in Listings 204 and 205; notice that in Listing 204 neither of the special code blocks have received indentation, while in Listing 205 both code blocks have received a single space of indentation.
The function $f$ has formula
\[
  f(x) = x^2.
\]
If you like splitting dollars,
\[
  g(x) = f(2x)
\]

### 5.8.6 afterHeading code blocks

Let's use the example Listing 206 for demonstration throughout this Section. As discussed on page 46, by default `latexindent.pl` will not add indentation after headings.

\[
\text{\texttt{\textbackslash paragraph\{paragraph title\}}}
\text{paragraph text}
\text{paragraph text}
\]

On using the YAML file in Listing 208 by running the command
```
$ latexindent.pl headings2.tex -l headings3.yaml
```
we obtain the output in Listing 207. Note that the argument of paragraph has received (default) indentation, and that the body after the heading statement has received (default) indentation.

\[
\text{\texttt{\textbackslash paragraph\{paragraph title\}}}
\text{paragraph text}
\text{paragraph text}
\]

If we specify `noAdditionalIndent` as in Listing 210 and run the command
```
$ latexindent.pl headings2.tex -l headings4.yaml
```
then we receive the output in Listing 209. 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.

\[
\text{\texttt{\textbackslash paragraph\{paragraph title\}}}
\text{paragraph text}
\text{paragraph text}
\]

Similarly, if we specify `indentRules` as in Listing 212 and run analogous commands to those above, we receive the output in Listing 211; 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 214 which gives the output in Listing 213.

Analogously, we may specify `indentRules` as in Listing 216 which gives the output in Listing 215; 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 218 and 220 respectively, with respective output in Listings 217 and 219. Note that in Listing 218 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 219, the argument has received both a (default) tab plus two spaces of indentation (from the global rule specified in Listing 220), and the remaining body after `paragraph` has received just two spaces of indentation.
5.8.7 The remaining code blocks

Referencing the different types of code blocks in Table 2 on page 49, we have a few code blocks yet to cover; these are very similar to the commands code block type covered comprehensively in Section 5.8.3 on page 57, but a small discussion defining these remaining code blocks is necessary.

5.8.7.1 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 2 on page 49;
- 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 511 on page 132

An example is shown in Listing 221, with the default output given in Listing 222.

In Listing 222, note that the maximum indentation is three tabs, and these come from:

- the \latexindent 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 48.

5.8.7.2 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 2 on page 49;
- 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 511 on page 132

A simple example is given in Listing 223, with default output in Listing 224.

In particular, latexindent.pl considers child, parent and node all to be namedGroupingBracesBrackets. Referencing Listing 224, note that the maximum indentation is two tabs, and these come from:

\footnote{You may like to verify this by using the -tt option and checking indent.log!}

\begin{Verbatim}
\begin{lstlisting}[language=tex]
\coordinate
child[grow=down]{
edge from parent [antiparticle]
node [above=3pt] {$C$}
}
\end{lstlisting}
\end{Verbatim}
• 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 48.

5.8.7.3 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 511 on page 132

An example is shown in Listing 225 with default output given in Listing 226.

Referencing Listing 226, 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 48.

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

5.8.7.4 filecontents

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

5.8.8 Summary

Having considered all of the different types of code blocks, the functions of the fields given in Listings 227 and 228 should now make sense.
5.9 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 229.

<table>
<thead>
<tr>
<th>listing 229: 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>- '+$'</td>
</tr>
<tr>
<td>- '--'</td>
</tr>
<tr>
<td>- '#\d'</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 230.

<table>
<thead>
<tr>
<th>Listing 230: pstricks1.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>\defFunction[algebraic]{torus}(u,v)</td>
</tr>
<tr>
<td>{(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>

<table>
<thead>
<tr>
<th>Listing 231: pstricks1 default output</th>
</tr>
</thead>
<tbody>
<tr>
<td>\defFunction[algebraic]{torus}(u,v)</td>
</tr>
<tr>
<td>{(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 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 229, then latexindent.pl will allow round parenthesis between optional and mandatory arguments. In the case of the code in Listing 230, latexindent.pl finds all the arguments of defFunction, both before and after (u,v).

The default output from running latexindent.pl on Listing 230 actually leaves it unchanged (see Listing 231); note in particular, this is because of noAdditionalIndentGlobal as discussed on page 59.

Upon using the YAML settings in Listing 233, and running the command

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

we obtain the output given in Listing 232.
5.9 Commands and the strings between their arguments

Listing 232: pstricks1.tex using Listing 233
\defFunction[algebraic]{torus}(u,v)
{(2+\cos(u))*\cos(v+\Pi)}
{(2+\cos(u))*\sin(v+\Pi)}
{\sin(u)}

Listing 233: noRoundParentheses.yaml
commandCodeBlocks:
  roundParenthesesAllowed: 0

Notice the difference between Listing 231 and Listing 232; in particular, in Listing 232, 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 2 on page 49) which, by default, assume indentation for their body, and hence the tabbed indentation in Listing 232.

Let’s explore this using the YAML given in Listing 235 and run the command

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

then the output is as in Listing 234.

Listing 234: pstricks1.tex using Listing 235
\defFunction[algebraic]{torus}(u,v)
\{(2+\cos(u))*\cos(v+\Pi)}
\{(2+\cos(u))*\sin(v+\Pi)}
\{\sin(u)}

Listing 235: defFunction.yaml
indentRules:
defFunction:
  body: " "

Notice in Listing 234 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 235.

stringsAllowedBetweenArguments: (fields)

tikz users may well specify code such as that given in Listing 236; processing this code using latexindent.pl gives the default output in Listing 237.

Listing 236: tikz-node1.tex
\draw[thin]
(c) to[in=110,out=-90]
++(0,-0.5cm)
node[below,align=left,scale=0.5]

Listing 237: 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 229 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 236, 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
5.9 Commands and the strings between their arguments

- the string \texttt{node} (specified in \texttt{stringsAllowedBetweenArguments})
- the optional argument [\texttt{below,align=left,scale=0.5}]

We can explore this further, for example using Listing 239 and running the command

\texttt{cmh:~$ latexindent.pl tikz-node1.tex -l draw.yaml}

we receive the output given in Listing 238.

\textbf{Listing 238: tikz-node1.tex using Listing 239}

\begin{verbatim}
\draw[thin]
\(\texttt{(c)}\) to[in=110,out=-90]
++(0,-0.5cm)
node[\texttt{below,align=left,scale=0.5}]
\end{verbatim}

Notice that each line after the \texttt{draw} command (its 'body') in Listing 238 has been given the appropriate two-spaces worth of indentation specified in Listing 239.

Let's compare this with the output from using the YAML settings in Listing 241, and running the command

\texttt{cmh:~$ latexindent.pl tikz-node1.tex -l no-strings.yaml}

given in Listing 240.

\textbf{Listing 240: tikz-node1.tex using Listing 241}

\begin{verbatim}
\draw[thin]
\(\texttt{(c)}\) to[in=110,out=-90]
++(0,-0.5cm)
node[\texttt{below,align=left,scale=0.5}]
\end{verbatim}

In this case, \texttt{latexindent.pl} sees that:

- the \texttt{\draw} command finishes after the \texttt{(c)}, as \texttt{stringsAllowedBetweenArguments} has been set to 0 so there are no strings allowed between arguments;
- it finds a \texttt{namedGroupingBracesBrackets} called to (see Table 2 on page 49) with argument \texttt{[in=110,out=-90]}
- it finds another \texttt{namedGroupingBracesBrackets} but this time called node with argument \texttt{[below,align=left,scale=0.5]}

Referencing Listing 229 on page 66, we see that the first field in the \texttt{stringsAllowedBetweenArguments} is \texttt{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 242 or Listing 243 is equivalent to using the settings in Listing 244.
5.9 Commands and the strings between their arguments

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 245 means that only the strings specified in that field will be used.

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 243 to 245.

We may explore this feature further with the code in Listing 246, whose default output is given in Listing 247.

Let's compare this with the output from using the YAML settings in Listing 249, and running the command

```bash
$ latexindent -l foreach.yaml
```

given in Listing 248.

You might like to compare the output given in Listing 247 and Listing 248. Note, in particular, in Listing 247 that the `foreach` command has not included any of the subsequent strings, and that the braces have been treated as a `namedGroupingBracesBrackets`. In Listing 248 the `foreach` command has been allowed to have `\x/\y` and in between arguments because of the settings given.
in Listing 249.

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 250, which has default output in Listing 251.

Notice that in Listing 251 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 115.

For demonstration, we can compare this output with that given in Listing 252 in which the settings from Listing 253 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.

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 243 to 245.
SECTION 6

The -m (modifylinebreaks) switch

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

6.1 Text Wrapping .................................................. 72
   6.1.1 Text wrap quick start ................................. 73
   6.1.2 textWrapOptions: modifying line breaks by text wrapping .......... 73
   6.1.3 Text wrapping on a per-code-block basis ....................... 76
6.2 removeParagraphLineBreaks: modifying line breaks for paragraphs ............ 82
6.3 Combining removeParagraphLineBreaks and textWrapOptions .................... 88
   6.3.1 text wrapping beforeFindingChildCodeBlocks .................... 89
6.4 Summary of text wrapping ..................................... 91
6.5 oneSentencePerLine: modifying line breaks for sentences ....................... 91
   6.5.1 sentencesFollow ......................................... 93
   6.5.2 sentencesBeginWith .................................... 94
   6.5.3 sentencesEndWith ....................................... 95
   6.5.4 Features of the oneSentencePerLine routine .................... 97
   6.5.5 Text wrapping and indenting sentences .......................... 98
6.6 Poly-switches .................................................. 100
   6.6.1 Poly-switches for environments .............................. 100
      6.6.1.1 Adding line breaks: BeginStartsOnOwnLine and BodyStartsOnOwn-
               Line ..................................................... 101
      6.6.1.2 Adding line breaks using EndStartsOnOwnLine and EndFinishesWith-
               LineBreak .............................................. 103
      6.6.1.3 poly-switches 1, 2, and 3 only add line breaks when necessary . 104
      6.6.1.4 Removing line breaks (poly-switches set to −1) ............... 105
      6.6.1.5 About trailing horizontal space .......................... 106
      6.6.1.6 poly-switch line break removal and blank lines ............... 106
   6.6.2 Poly-switches for double back slash ................................ 108
      6.6.2.1 Double back slash starts on own line ....................... 108
      6.6.2.2 Double back slash finishes with line break ................. 109
      6.6.2.3 Double back slash poly-switches for specialBeginEnd .......... 109
      6.6.2.4 Double back slash poly-switches for optional and mandatory arguments110
      6.6.2.5 Double back slash optional square brackets .................. 111
   6.6.3 Poly-switches for other code blocks ................................ 111
   6.6.4 Partnering BodyStartsOnOwnLine with argument-based poly-switches .......... 113
6.1 Text Wrapping

There are many different configuration options for the text wrapping routine of \texttt{latexindent.pl}, perhaps too many. The following sections are comprehensive, but quite long; in an attempt to be brief, you might begin with the settings given in Section 6.1.1.

---

6.1 Text Wrapping

There are many different configuration options for the text wrapping routine of \texttt{latexindent.pl}, perhaps too many. The following sections are comprehensive, but quite long; in an attempt to be brief, you might begin with the settings given in Section 6.1.1.

---

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 254.

**LISTING 254: modifyLineBreaks**

```
486 modifyLineBreaks:
487   preserveBlankLines: 1
488   condenseMultipleBlankLinesInto: 1
```

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:

*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.*

**preserveBlankLines**: 0|1

This field is directly related to *poly-switches*, discussed in Section 6.6. 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.

**condenseMultipleBlankLinesInto**: (positive integer)

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 255 shows a sample file with blank lines; upon running

```
cmh:~$ latexindent.pl myfile.tex -m -o=+-mod1
```

the output is shown in Listing 256; note that the multiple blank lines have been condensed into one blank line, and note also that we have used the \texttt{-m} switch!

**LISTING 255: mlb1.tex**

```
before blank line
after blank line
after blank line
```

**LISTING 256: mlb1-mod1.tex**

```
before blank line
after blank line
```

---

6.6.5 Conflicting poly-switches: sequential code blocks

6.6.6 Conflicting poly-switches: nested code blocks
6.1.1 Text wrap quick start

Of all the available text wrapping options, I consider Listing 257 to be among the most helpful starting points.

**Listing 257: textwrap-qs.yaml**

```yaml
modifyLineBreaks:
  textWrapOptions:
    columns: 80 # number of columns
    perCodeBlockBasis: 1 # per-code-block wrap
    beforeFindingChildCodeBlocks: 1 # wrap *before* finding child code blocks
    mainDocument: 1 # apply to main document
    afterHeading: 1 # after headings
    items: 1 # within items
    removeParagraphLineBreaks: # remove line breaks within paragraphs
      mainDocument: 1
      afterHeading: 1
      items: 1
    beforeTextWrap: 1 # before wrapping text
```

You can read about `perCodeBlockBasis` in Section 6.1.3 and `removeParagraphLineBreaks` in Section 6.2.

If the settings in Listing 257 do not give your desired output, take a look at the demonstration in Section 6.3.1, in particular Listing 324.

6.1.2 textWrapOptions: modifying line breaks by text wrapping

When the `-m` switch is active `latexindent.pl` has the ability to wrap text using the options specified in the `textWrapOptions` field, see Listing 258.

**Listing 258: textWrapOptions**

```yaml
513  textWrapOptions:
514    columns: 0
```

The value of `columns` specifies the column at which the text should be wrapped.

By default, the value of `columns` is 0, so `latexindent.pl` will 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.

By default, the text wrapping routine will operate *before* the code blocks have been searched for; text wrapping on a *per-code-block* basis is discussed in Section 6.1.3.

We consider the file given in Listing 259 for demonstration.

**Listing 259: 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 261, and running the command

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

we obtain the output in Listing 260.
6.1 Text Wrapping

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

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

The text wrapping routine is performed \emph{after} verbatim environments have been stored, so verbatim environments and verbatim commands are exempt from the routine. For example, using the file in Listing 262,

```
\begin{verbatim}
a long line in a verbatim environment, which will not be broken by latexindent.pl
\end{verbatim}
```

Here is a verb command: \texttt{\verb!this will not be text wrapped!}

and running the following command and continuing to use \texttt{textwrap1.yaml} from Listing 261,

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

then the output is as in Listing 263.

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 264

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

Here is a line \% text wrapping does not apply to comments by latexindent.pl
and running the following command and continuing to use textwrap1.yaml from Listing 261,

```
cmh:~$ latexindent.pl -m textwrap3.tex -o textwrap3-mod1.tex -l textwrap1.yaml
```

then the output is as in Listing 265.

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

Here is a line
% text wrapping does not apply to comments by latexindent.pl
```

The default value of huge is overflow, which means that words will not be broken by the text wrapping routine, implemented by the Text::Wrap [26]. There are options to change the huge option for the Text::Wrap module to either wrap or die. Before modifying the value of huge, please bear in mind the following warning:

⚠️ Changing the value of huge to anything other than overflow will slow down latexindent.pl significantly when the -m switch is active. Furthermore, changing huge means that you may have some words or commands(!) split across lines in your .tex file, which may affect your output. I do not recommend changing this field.

For example, using the settings in Listings 267 and 269 and running the commands

```
cmh:~$ latexindent.pl -m textwrap4.tex -o=+-mod2A -l textwrap2A.yaml
cmh:~$ latexindent.pl -m textwrap4.tex -o=+-mod2B -l textwrap2B.yaml
```

gives the respective output in Listings 266 and 268.

```
Listing 266: textwrap4-mod2A.tex
Here is a line of text.
```

```
Listing 267: textwrap2A.yaml
modifyLineBreaks:
  textWrapOptions:
    columns: 3
    huge: wrap
```

```
Listing 268: textwrap4-mod2B.tex
Here is a line of text.
```

```
Listing 269: textwrap2B.yaml
modifyLineBreaks:
  textWrapOptions:
    columns: 3
```

You can also specify the tabstop field as an integer value, which is passed to the text wrap module; see [26] for details. Starting with the code in Listing 270 with settings in Listing 271, and running
the command

```bash
$ latexindent.pl -m textwrap-ts.tex -o=-mod1 -l tabstop.yaml
```

gives the code given in Listing 272.

You can specify separator, break and unexpand options in your settings in analogous ways to those demonstrated in Listings 269 and 271, and they will be passed to the `Text::Wrap` module. I have not found a useful reason to do this; see [26] for more details.

### 6.1.3 Text wrapping on a per-code-block basis

By default, if the value of `columns` is greater than 0 and the `-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 `latexindent.pl` to apply `textWrap` on a per-code-block basis. Thanks to [30] for their help in testing and shaping this feature.

The full details of `textWrapOptions` are shown in Listing 273. In particular, note the field `perCodeBlockBasis`.

The code blocks detailed in Listing 273 are with direct reference to those detailed in Table 2 on page 49.

The only special case is the `mainDocument` 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 292 on page 82; in fact, the two features can even be combined (this is detailed in Section 6.3 on page 88).

Note: `mainDocument` replaces `masterDocument` which was used in previous versions of `latexindent.pl`. The field `masterDocument` is still supported, but it is anticipated to be removed in a future version, so I recommend using `mainDocument` instead.

Let’s explore these switches with reference to the code given in Listing 274; the text outside of the environment is considered part of the `mainDocument`. 
6.1 Text Wrapping

LISTING 274: textwrap5.tex

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}

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 275 to 277 each give the same output.

Let’s explore the similarities and differences in the equivalent (with respect to Listing 274) syntax specified in Listings 275 to 277:

- in each of Listings 275 to 277 notice that columns: 30;
- in each of Listings 275 to 277 notice that perCodeBlockBasis: 1;
- in Listing 275 we have specified all: 1 so that the text wrapping will operate upon all code blocks;
- in Listing 276 we have not specified all, and instead, have specified that text wrapping should be applied to each of environments and mainDocument;
- in Listing 277 we have specified text wrapping for mainDocument and on a per-name basis for environments code blocks.

Upon running the following commands

cmh:~$ latexindent.pl -s textwrap5.tex -l=textwrap3.yaml -m

we obtain the output shown in Listing 278.

LISTING 278: textwrap5-mod3.tex

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}

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 277 by using Listing 279.
6.1 Text Wrapping

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.

In particular, upon running

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

we obtain the output given in Listing 280.

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 277) 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 279 in the settings given in Listings 281 to 283.

Upon running the commands

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

we obtain the output given in Listing 280.

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 277) 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 279 in the settings given in Listings 281 to 283.

Upon running the commands

\texttt{cmh:~$ latexindent.pl -s textwrap6.tex -l=textwrap5.yaml -m}
6.1 Text Wrapping

we receive the respective output given in Listings 284 to 286.

---

**Listing 284: textwrap6.tex using Listing 281**

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.

---

**Listing 285: textwrap6.tex using Listing 282**

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.
6.1 Text Wrapping

LISTING 286: textwrap6.tex using Listing 283

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:

- in Listing 284 the text wrapping routine has not been applied to any environments because it has been switched off (per-code-block) in Listing 281;
- in Listing 285 the text wrapping routine has not been applied to myenv because it has been switched off (per-name) in Listing 282;
- in Listing 286 the text wrapping routine has not been applied to mainDocument because of the settings in Listing 283.

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 279 on page 78, for example). We explore further options in Listings 287 to 289.

Listing 287 and Listing 288 are equivalent. Upon running the commands

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

we receive the respective output given in Listings 290 and 291.
6.1 Text Wrapping

Listing 290: textwrap6.tex using Listing 287

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.

Listing 291: textwrap6.tex using Listing 289

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:

- in Listing 290 the text for the mainDocument has been wrapped using 30 columns, while environments has been wrapped using 50 columns;
- in Listing 291 the text for myenv has been wrapped using 50 columns, the text for another has been wrapped using 15 columns, and mainDocument 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 33 on page 30 then text wrapping will be disabled for this code block.

If you wish to specify afterHeading commands (see Listing 122 on page 46) on a per-name basis, then you need to append the name with :heading, for example, you might use section:heading.
6.2 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 292. Thank you to [19] for shaping and assisting with the testing of this feature.

This feature is considered complimentary to the oneSentencePerLine feature described in Section 6.5 on page 91.

This routine can be turned on globally for every code block type known to latexindent.pl (see Table 2 on page 49) 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 2 on page 49;
- 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 309 on page 87.

Let’s start with the .tex file in Listing 293, together with the YAML settings in Listing 294.

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 295.

**LISTING 295: shortlines1.tex**

\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 293 which has crept into the output in Listing 295. This can be fixed using the YAML file in Listing 418 on page 106 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 296; notice that the double spaces present in Listing 295 have been addressed.

**LISTING 296: shortlines1-tws.tex**

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

    Another paragraph.
\end{myenv}

Keeping with the settings in Listing 294, we note that the all switch applies to all code block types. So, for example, let’s consider the files in Listings 297 and 298

**LISTING 297: shortlines-mand.tex**

```
\mycommand{
    The lines in this command are very short and contain many linebreaks.

    Another paragraph.
}
```

**LISTING 298: shortlines-opt.tex**

```
\mycommand[
    The lines in this command are very short and contain many linebreaks.

    Another paragraph.
]
```

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 299 and 300.

**LISTING 299: shortlines-mand1.tex**

```
\mycommand{
    The lines in this command are very short and contain many linebreaks.

    Another paragraph.
}
```
6.2 removeParagraphLineBreaks: modifying line breaks for paragraphs

Listing 300: shortlines-opt1.tex

\mycommand{

The lines in this command are very short and contain many linebreaks.

Another paragraph.
}

Listing 301: shortlines-envs.tex

\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}

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-type basis, or on a per-name basis.

For example, let's use the code in Listing 301, and consider the settings in Listings 302 and 303; note that in Listing 302 we specify that every environment should receive treatment from the routine, while in Listing 303 we specify that only the one environment should receive the treatment.

Listing 302: remove-para2.yaml

modifyLineBreaks:
removeParagraphLineBreaks:
environments: 1

Listing 303: remove-para3.yaml

modifyLineBreaks:
removeParagraphLineBreaks:
environments:
one: 1

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 304 and 305.
LISTING 304: shortlines-envs2.tex

\begin{one}
\begin{itemize}
\item The lines in this environment are very short and contain many linebreaks.
\item Another paragraph.
\end{itemize}
\end{one}

\begin{two}
\begin{itemize}
\item The lines in this environment are very short and contain many linebreaks.
\item Another paragraph.
\end{itemize}
\end{two}

LISTING 305: shortlines-envs3.tex

\begin{one}
\begin{itemize}
\item The lines in this environment are very short and contain many linebreaks.
\item Another paragraph.
\end{itemize}
\end{one}

\begin{two}
\begin{itemize}
\item The lines in this environment are very short and contain many linebreaks.
\item Another paragraph.
\end{itemize}
\end{two}

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 mainDocument 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 306, with the YAML settings in Listing 307.

Note: mainDocument replaces masterDocument which was used in previous versions of latexindent.pl. The field masterDocument is still supported, but it is anticipated to be removed in a future version, so I recommend using mainDocument instead.
6.2 removeParagraphLineBreaks: modifying line breaks for paragraphs

The lines in this document are very short and contain many linebreaks.

Another paragraph.

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

Upon running the following command

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

then we obtain the output in Listing 308.

```
The lines in this document are very short and contain many linebreaks.

Another paragraph.

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

Note that the all field can take the same exceptions detailed in Listings 281 to 283.

```
paragraphsStopAt: (fields)
```

The paragraph line break routine considers blank lines and the \par command to be the end of a paragraph; you can fine tune the behaviour of the routine further by using the paragraphsStopAt fields, shown in Listing 309.
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 309. 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 310; we will, in turn, consider the settings in Listings 311 and 312.

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

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

Upon using the settings from Listing 307 on the previous page and running the commands

```
cmh:~$ latexindent.pl -m sl-stop.tex -o sl-stop4.tex -l remove-para4.yaml
```

```
cmh:~$ latexindent.pl -m sl-stop.tex -o sl-stop4-comment.tex -l=remove-para4.yaml,stop-comment.yaml
```

we obtain the respective outputs in Listings 313 to 315; notice in particular that:

- in Listing 313 the paragraph line break routine has included commands and comments;
- in Listing 314 the paragraph line break routine has stopped at the `\texttt{emph}` command, because in Listing 311 we have specified `commands` to be 1, and `\texttt{emph}` is at the beginning of a line;
- in Listing 315 the paragraph line break routine has stopped at the comments, because in Listing 312 we have specified `comments` to be 1, and the comment is at the beginning of a line.

In all outputs in Listings 313 to 315 we notice that the paragraph line break routine has stopped at `\begin{myenv}` because, by default, environments is set to 1 in Listing 309.
6.3 Combining removeParagraphLineBreaks and textWrapOptions

The text wrapping routine (Section 6.1 on page 72) and remove paragraph line breaks routine (Section 6.2 on page 82) can be combined.

We motivate this feature with the code given in Listing 316.

```
LISTING 316: textwrap7.tex

This paragraph has line breaks throughout its paragraph; we would like to combine the textwrapping and paragraph removal routine.

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 319 and running the command

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

we obtain the output in Listing 318.
6.3 Combining removeParagraphLineBreaks and textWrapOptions

<table>
<thead>
<tr>
<th>Listing 318: textwrap7-mod12.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>This paragraph has line breaks throughout its paragraph; we would like to combine the textwrapping and paragraph removal routine.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 319: textwrap12.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>modifyLineBreaks:</td>
</tr>
<tr>
<td>textWrapOptions:</td>
</tr>
<tr>
<td>columns: 30</td>
</tr>
<tr>
<td>perCodeBlockBasis: 1</td>
</tr>
<tr>
<td>all: 1</td>
</tr>
<tr>
<td>removeParagraphLineBreaks:</td>
</tr>
<tr>
<td>all: 1</td>
</tr>
<tr>
<td>beforeTextWrap: 1</td>
</tr>
</tbody>
</table>

In Listing 318 the paragraph line breaks have first been removed from Listing 316, and then the text wrapping routine has been applied. It is envisaged that variants of Listing 319 will be among the most useful settings for these two features.

6.3.1 text wrapping beforeFindingChildCodeBlocks

I think it likely that most users will wish to employ the beforeFindingChildCodeBlocks option for the text wrap routine.

To motivate its use, we begin with the file in Listing 320.

<table>
<thead>
<tr>
<th>Listing 320: textwrap-bfccb.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>one</td>
</tr>
<tr>
<td>two three four \text{test}</td>
</tr>
<tr>
<td>five six seven</td>
</tr>
<tr>
<td>eight nine}</td>
</tr>
<tr>
<td>ten eleven</td>
</tr>
<tr>
<td>twelve thirteen</td>
</tr>
<tr>
<td>fourteen fifteen sixteen seventeen</td>
</tr>
</tbody>
</table>

Using the settings in Listing 319 and running

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

gives the output in Listing 321

<table>
<thead>
<tr>
<th>Listing 321: textwrap-bfccb-mod12.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>one</td>
</tr>
<tr>
<td>two three four</td>
</tr>
<tr>
<td>\text{test five six seven eight}</td>
</tr>
<tr>
<td>nine}</td>
</tr>
<tr>
<td>ten</td>
</tr>
<tr>
<td>eleven twelve thirteen</td>
</tr>
<tr>
<td>fourteen fifteen sixteen seventeen</td>
</tr>
</tbody>
</table>

Note that we have added a ‘ruler’ to Listing 321 to assist with measuring.

The output in Listing 321 is not ideal, but it is expected. The reasoning is as follows:

- latexindent.pl first of all searches for code blocks (see Table 2 on page 49);
- it replaces each code block with a unique identifying string;
- with the settings of Listing 319 in place, it performs the paragraph line break removal, and then the text wrapping routine first of all on the text command, and then on the surrounding text;
- the surrounding text does not know that text is a command.

We can instruct latexindent.pl to perform text wrapping before searching for child code blocks by using the beforeFindingChildCodeBlocks field.
We save the quick-start settings from Listing 257 into Listing 322 and change the value of columns for demonstration. Upon running the command

```
cmh:~$ latexindent.pl -m textwrap-bfccb.tex -l=textwrap13.yaml -o=-mod13
```

we receive the output in Listing 323.

```
<table>
<thead>
<tr>
<th>one</th>
<th>two</th>
<th>three</th>
<th>four</th>
<th>\text{test}</th>
<th>five</th>
<th>six</th>
<th>seven</th>
<th>eight</th>
<th>nine</th>
</tr>
</thead>
</table>

This output is different from Listing 321, but is still not ideal, as the test command has indented its mandatory argument. We can employ noAdditionalIndent from Section 5.8 on page 48 in Listing 325 and run the command

```
cmh:~$ latexindent.pl -m textwrap-bfccb.tex -l=textwrap14.yaml -o=-mod14
```

to receive the output in Listing 324.

For reference, let’s say that we had started from Listing 319, which instructs latexindent.pl to apply the text-wrapping and paragraph-line-break-removal routines to all code blocks. In order to
achieve the output in Listing 324, then we would need to employ an exception, which we demonstrate in Listing 326.

**Listing 326: textwrap15.yaml**

```yaml
modifyLineBreaks:
  textWrapOptions:
    columns: 40
    perCodeBlockBasis: 1
    beforeFindingChildCodeBlocks: 1
    all: 1
  removeParagraphLineBreaks:
    all:
      except:
      - test
    beforeTextWrap: 1

noAdditionalIndent:
  test: 1
```

### 6.4 Summary of text wrapping

I consider the most useful starting point for text wrapping to be given in Section 6.1.1 and Section 6.3.1.

Starting from Listing 257, it is likely that you will have to experiment with making adjustments (such as that given in Listing 325) depending on your preference.

It is important to note the following:

- verbatim code blocks of all types will *not* be affected by the text wrapping routine. See the demonstration in Listing 263 on page 74, together with environments: Listing 18 on page 27, commands: Listing 19 on page 27, noIndentBlock: Listing 20, specialBeginEnd: Listing 116 on page 44;
- comments will *not* be affected by the text wrapping routine (see Listing 265 on page 75);
- it is possible to wrap text on a per-code-block and a per-name basis;
- indentation is performed *after* the text wrapping routine; as such, indented code will likely exceed any maximum value set in the columns field.

### 6.5 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 [17] for helping to shape and test this feature. The behaviour of this part of the script is controlled by the switches detailed in Listing 327, all of which we discuss next.
This is a binary switch that details if \texttt{latexindent.pl} should perform the sentence manipulation routine; it is \textit{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.

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

For example, consider \texttt{multiple-sentences.tex} shown in Listing 328.

If we use the YAML files in Listings 330 and 332, and run the commands

\begin{verbatim}
$ latexindent.pl multiple-sentences -m -l=manipulate-sentences.yaml
$ latexindent.pl multiple-sentences -m -l=keep-sen-line-breaks.yaml
\end{verbatim}

then we obtain the respective output given in Listings 329 and 331.
6.5 oneSentencePerLine: modifying line breaks for sentences

- **Listing 329: multiple-sentences.tex**
  using Listing 330

  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.

- **Listing 331: multiple-sentences.tex**
  using Listing 332

  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, in particular, that the ‘internal’ sentence line breaks in Listing 328 have been removed in Listing 329, but have not been removed in Listing 331.

The remainder of the settings displayed in Listing 327 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 333); 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 334); by default, this is only capital letters;
- **end** with a character (see Listing 335); 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.

- **Listing 333: sentencesFollow**

  sentencesFollow:
  par: 1
  blankLine: 1
  fullStop: 1
  exclamationMark: 1
  questionMark: 1
  rightBrace: 1
  commentOnPreviousLine: 1
  other: 0

- **Listing 334: sentencesBeginWith**

  sentencesBeginWith:
  A-Z: 1
  a-z: 0
  other: 0

- **Listing 335: sentencesEndWith**

  sentencesEndWith:
  basicFullStop: 0
  betterFullStop: 1
  exclamationMark: 1
  questionMark: 1
  other: 0

---

### 6.5.1 sentencesFollow

Let’s explore a few of the switches in sentencesFollow; let’s start with Listing 328 on the previous page, and use the YAML settings given in Listing 337. Using the command

```
cmh:~$ latexindent.pl multiple-sentences -m -l=sentences-follow1.yaml
```
we obtain the output given in Listing 336.

Listing 336: multiple-sentences.tex using Listing 337

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 333 with the .tex file detailed in Listing 338.

Listing 338: 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

```
cmh:~$ latexindent.pl multiple-sentences1 -m -l=manipulate-sentences.yaml
```

then we obtain the respective output given in Listings 339 and 340.

Listing 339: multiple-sentences1.tex using Listing 330 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 340: multiple-sentences1.tex using Listing 341

(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 339 the first sentence after the ) has not been accounted for, but that following the inclusion of Listing 341, the output given in Listing 340 demonstrates that the sentence has been accounted for correctly.

6.5.2 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 334), and we can use the other field to define sentences to begin with other characters.
6.5 oneSentencePerLine: modifying line breaks for sentences

This is the first sentence.

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

Upon running the following commands

```
cmh:$ latexindent.pl multiple-sentences2 -m -l=manipulate-sentences.yaml

cmh:$ latexindent.pl multiple-sentences2 -m -l=manipulate-sentences.yaml,sentences-begin1.yaml
```

then we obtain the respective output given in Listings 343 and 344.

Listing 343: multiple-sentences2.tex using Listing 330 on page 93

This is the first sentence.

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

Listing 344: multiple-sentences2.tex using Listing 345

This is the first sentence.

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

Notice that in Listing 343, the first sentence has been accounted for but that the subsequent sentences have not. In Listing 344, all of the sentences have been accounted for, because the other field in Listing 345 has defined sentences to begin with either $ or any numeric digit, 0 to 9.

6.5.3 sentencesEndWith

Let's return to Listing 328 on page 92; we have already seen the default way in which latexindent.pl will operate on the sentences in this file in Listing 329 on page 93. We can populate the other field with any character that we wish; for example, using the YAML specified in Listing 347 and the command

```
cmh:$ latexindent.pl multiple-sentences -m -l=sentences-end1.yaml

cmh:$ latexindent.pl multiple-sentences -m -l=sentences-end2.yaml
```

then we obtain the output in Listing 346.

Listing 346: multiple-sentences.tex using Listing 347

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.
6.5 oneSentencePerLine: modifying line breaks for sentences

There is a subtle difference between the output in Listings 346 and 348; in particular, in Listing 346 the word `sentence` has not been defined as a sentence, because we have not instructed `latexindent.pl` to begin sentences with lower case letters. We have changed this by using the settings in Listing 349, and the associated output in Listing 348 reflects this.

Referencing Listing 335 on page 93, you’ll notice that there is a field called `basicFullStop`, which is set to 0, and that the `betterFullStop` is set to 1 by default.

Let’s consider the file shown in Listing 350.

Upon running the following commands

```
cmh:~$ latexindent.pl url -m -l=manipulate-sentences.yaml
```

we obtain the output given in Listing 351.

```
LISTING 351: url.tex using Listing 330 on page 93
This sentence, \url{tex.stackexchange.com/} finishes here. Second sentence.
```

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

- they are ignored within e.g. and 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 `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 `betterFullStop` routine by using the `fine tuning`, detailed in Listing 511 on page 132.

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

```
cmh:~$ latexindent.pl url -m -l=alt-full-stop1.yaml
```

and the YAML in Listing 353 gives the output in Listing 352.
6.5 oneSentencePerLine: modifying line breaks for sentences

Notice that the full stop within the URL has not been accommodated correctly because of the non-default settings in Listing 353.

6.5.4 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 354, and run the command

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

then we obtain the output in Listing 355.

```latex
\begin{verbatim}
there are sentences within this. These will not be operated upon by latexindent.pl.
\end{verbatim}
and finishes here. Second sentence \% a commented full stop. contains trailing comments, which are ignored.
```

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 356 and run the commands

```bash
$ latexindent.pl multiple-sentences4 -m -l=manipulate-sentences.yaml
$ latexindent.pl multiple-sentences4 -m -l=keep-sen-line-breaks.yaml
```

then we obtain the output in Listings 357 and 358.
6.5 oneSentencePerLine: modifying line breaks for sentences

<table>
<thead>
<tr>
<th>Listing 356: multiple-sentences4.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>This sentence \begin{itemize}</td>
</tr>
<tr>
<td>\item continues \end{itemize}</td>
</tr>
<tr>
<td>across itemize and finishes here.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 357: multiple-sentences4.tex using Listing 330 on page 93</th>
</tr>
</thead>
<tbody>
<tr>
<td>This sentence \begin{itemize} \item continues \end{itemize} across itemize and finishes here.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 358: multiple-sentences4.tex using Listing 332 on page 93</th>
</tr>
</thead>
<tbody>
<tr>
<td>This sentence \begin{itemize} \item continues \end{itemize} across itemize and finishes here.</td>
</tr>
</tbody>
</table>

Once you've read Section 6.6, you will know that you can accommodate the removal of internal sentence line breaks by using the YAML in Listing 360 and the command

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

the output of which is shown in Listing 359.

<table>
<thead>
<tr>
<th>Listing 359: multiple-sentences4.tex using Listing 360</th>
</tr>
</thead>
<tbody>
<tr>
<td>This sentence \begin{itemize} \item continues \end{itemize} across itemize and finishes here.</td>
</tr>
</tbody>
</table>

### 6.5.5 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 361.

<table>
<thead>
<tr>
<th>Listing 361: multiple-sentences5.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>A distinção entre conteúdo \textit{real} e conteúdo \textit{intencional} está relacionada, ainda, à distinção entre o conceito husserliano de \textit{experiência} e o uso popular desse termo. No sentido comum, o \textit{experimentado} é um complexo de eventos exteriores, e o \textit{experimentar} consiste em percepções (além de julgamentos e outros atos) nas quais tais eventos aparecem como objetos, e objetos frequentemente relacionados ao ego empírico.</td>
</tr>
</tbody>
</table>

Referencing Listing 363, and running the following command

```bash
cmh:~$ latexindent.pl multiple-sentences5 -m -l=sentence-wrap1.yaml
```

we receive the output given in Listing 362.
A distinção entre conteúdo \textit{real} e conteúdo \textit{intencional} está relacionada, ainda, à distinção entre o conceito husserliano de \textit{experiência} e o uso popular desse termo. No sentido comum, o \textit{experimentado} é um complexo de eventos exteriores, e o \textit{experimentar} consiste em percepções (além de julgamentos e outros atos) nas quais tais eventos aparecem como objetos, e objetos frequentemente relacionados ao ego empírico.

If you wish to specify the \texttt{columns} field on a per-code-block basis for sentences, then you would use \texttt{sentence}; explicitly, starting with Listing 287 on page 80, for example, you would replace/append environments with, for example, \texttt{sentence: 50}.

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

The indentation of sentences requires that sentences are stored as code blocks. This means that you may need to tweak Listing 335 on page 93. Let's explore this in relation to Listing 364.

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

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

\begin{verbatim}
cmh:\$ latexindent.pl multiple-sentences6 -m -l=sentence-wrap1.yaml
cmh:\$ latexindent.pl multiple-sentences6 -m -l=sentence-wrap1.yaml -y="modifyLineBreaks:oneSentencePerLine:sentenceIndent:"
\end{verbatim}

we receive the respective output in Listing 365 and Listing 366.

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

We note that Listing 365 the \texttt{itemize} code block has \texttt{not} been indented appropriately. This is because the \texttt{oneSentencePerLine} has been instructed to store sentences (because Listing 363); each sentence is then searched for code blocks.

We can tweak the settings in Listing 335 on page 93 to ensure that full stops are not followed by \texttt{item} commands, and that the end of sentences contains \texttt{\end{itemize}} as in Listing 367 (if you intend to use this, ensure that you remove the line breaks from the other field).
6.6 Poly-switches

Upon running

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

we receive the output in Listing 368.

Consider the following: \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.

### 6.6 Poly-switches

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

- **1** remove mode: line breaks before or after the `<part of thing>` can be removed (assuming that preserveBlankLines is set to 0);
- **0** off mode: line breaks will not be modified for the `<part of thing>` under consideration;
- **1** add mode: a line break will be added before or after the `<part of thing>` under consideration, assuming that there is not already a line break before or after the `<part of thing>`;
- **2** comment then add mode: a comment symbol will be added, followed by a line break before or after the `<part of thing>` under consideration, assuming that there is not already a comment and line break before or after the `<part of thing>`;
- **3** add then blank line mode: a line break will be added before or after the `<part of thing>` under consideration, assuming that there is not already a line break before or after the `<part of thing>`, followed by a blank line;
- **4** add blank line mode: a blank line will be added before or after the `<part of thing>` under consideration, even if the `<part of thing>` is already on its own line.

In the above, `<part of thing>` refers to either the begin statement, body or end statement of the code blocks detailed in Table 2 on page 49. 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.

#### 6.6.1 Poly-switches for environments

We start by viewing a snippet of defaultSettings.yaml in Listing 369; note that it contains global settings (immediately after the environments field) and that per-name settings are also allowed – in the case of Listing 369, settings for equation* have been specified for demonstration. Note that all poly-switches are off (set to 0) by default.
Let's begin with the simple example given in Listing 370; note that we have annotated key parts of the file using ♠, ♥, ♦ and ♣, these will be related to fields specified in Listing 369.

### Adding line breaks: BeginStartsOnOwnLine and BodyStartsOnOwnLine

Let's explore `BeginStartsOnOwnLine` and `BodyStartsOnOwnLine` in Listings 371 and 372, and in particular, let's allow each of them in turn to take a value of 1.

```yaml
Listing 371: env-mlb1.yaml
modifyLineBreaks:
environments:
  BeginStartsOnOwnLine: 1
```

```yaml
Listing 372: env-mlb2.yaml
modifyLineBreaks:
environments:
  BodyStartsOnOwnLine: 1
```

After running the following commands,

```
$ latexindent .pl -m env-mlb.tex -l env-mlb1.yaml
$ latexindent .pl -m env-mlb.tex -l env-mlb2.yaml
```

the output is as in Listings 373 and 374 respectively.

```
Listing 373: env-mlb.tex using Listing 371
\begin{myenv}body of myenv\end{myenv}
```

```
Listing 374: env-mlb.tex using Listing 372
body of myenv\end{myenv}
```

There are a couple of points to note:

- in Listing 373 a line break has been added at the point denoted by ♠ in Listing 370; no other line breaks have been changed;

- in Listing 374 a line break has been added at the point denoted by ♥ in Listing 370; furthermore, note that the `body` of `myenv` has received the appropriate (default) indentation.

Let's now change each of the 1 values in Listings 371 and 372 so that they are 2 and save them into `env-mlb3.yaml` and `env-mlb4.yaml` respectively (see Listings 375 and 376).

```yaml
Listing 375: env-mlb3.yaml
modifyLineBreaks:
environments:
  BeginStartsOnOwnLine: 2
```

```yaml
Listing 376: env-mlb4.yaml
modifyLineBreaks:
environments:
  BodyStartsOnOwnLine: 2
```

Upon running commands analogous to the above, we obtain Listings 377 and 378.

```
Listing 377: env-mlb.tex using Listing 375
\begin{myenv}body of myenv\end{myenv}
```

```
Listing 378: env-mlb.tex using Listing 376
body of myenv\end{myenv}
```
Note that line breaks have been added as in Listings 373 and 374, 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 371 and 372 so that they are 3 and save them into `env-mlb5.yaml` and `env-mlb6.yaml` respectively (see Listings 379 and 380).

```
Listing 379: env-mlb5.yaml
modifyLineBreaks:
environments:
    BeginStartsOnOwnLine: 3
```
```
Listing 380: env-mlb6.yaml
modifyLineBreaks:
environments:
    BodyStartsOnOwnLine: 3
```

Upon running commands analogous to the above, we obtain Listings 381 and 382.

```
Listing 381: env-mlb.tex using Listing 379
\begin{myenv}body of myenv\end{myenv} after words
```
```
Listing 382: env-mlb.tex using Listing 380
body of myenv\end{myenv} after words
```

Let’s now change each of the 1 values in Listings 379 and 380 so that they are 4 and save them into `env-beg4.yaml` and `env-body4.yaml` respectively (see Listings 383 and 384).

```
Listing 383: env-beg4.yaml
modifyLineBreaks:
environments:
    BeginStartsOnOwnLine: 4
```
```
Listing 384: env-body4.yaml
modifyLineBreaks:
environments:
    BodyStartsOnOwnLine: 4
```

We will demonstrate this poly-switch value using the code in Listing 385.

```
Listing 385: env-mlb1.tex
\begin{myenv}body of myenv\end{myenv} after words
```

Upon running the commands
```
cmh:∼$ latexindent.pl -m env-mlb1.tex -l env-beg4.yaml
cmh:∼$ latexindent.pl -m env-mlb1.tex -l env-body4.yaml
```
then we receive the respective outputs in Listings 386 and 387.

```
Listing 386: env-mlb1.tex using Listing 383
\begin{myenv}body of myenv\end{myenv} after words
```
```
Listing 387: env-mlb1.tex using Listing 384
body of myenv\end{myenv} after words
```

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

1. in Listing 386 a blank line has been inserted before the \begin statement, even though the \begin statement was already on its own line;
2. in Listing 387 a blank line has been inserted before the beginning of the body, even though it already began on its own line.
6.6.1.2 Adding line breaks using EndStartsOnOwnLine and EndFinishesWithLineBreak

Let's explore EndStartsOnOwnLine and EndFinishesWithLineBreak in Listings 388 and 389, and in particular, let's allow each of them in turn to take a value of 1.

Listing 388: env-mlb7.yaml
modifyLineBreaks:
  environments:
    EndStartsOnOwnLine: 1

Listing 389: env-mlb8.yaml
modifyLineBreaks:
  environments:
    EndFinishesWithLineBreak: 1

After running the following commands,

cmh:~$ latexindent.pl -m env-mlb.tex -l env-mlb7.yaml

the output is as in Listings 390 and 391.

Listing 390: env-mlb.tex using Listing 388
before words \begin{myenv}body of myenv
\end{myenv} after words

Listing 391: env-mlb.tex using Listing 389
before words \begin{myenv}body of myenv\end{myenv}
after words

There are a couple of points to note:

- in Listing 390 a line break has been added at the point denoted by ♦ in Listing 370 on page 101; no other line breaks have been changed and the \end{myenv} statement has not received indentation (as intended);
- in Listing 391 a line break has been added at the point denoted by ♣ in Listing 370 on page 101.

Let's now change each of the 1 values in Listings 388 and 389 so that they are 2 and save them into env-mlb9.yaml and env-mlb10.yaml respectively (see Listings 392 and 393).

Listing 392: env-mlb9.yaml
modifyLineBreaks:
  environments:
    EndStartsOnOwnLine: 2

Listing 393: env-mlb10.yaml
modifyLineBreaks:
  environments:
    EndFinishesWithLineBreak: 2

Upon running commands analogous to the above, we obtain Listings 394 and 395.

Listing 394: env-mlb.tex using Listing 392
before words \begin{myenv}body of myenv
\end{myenv} after words

Listing 395: env-mlb.tex using Listing 393
before words \begin{myenv}body of myenv\end{myenv}
after words

Note that line breaks have been added as in Listings 390 and 391, 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 388 and 389 so that they are 3 and save them into env-mlb11.yaml and env-mlb12.yaml respectively (see Listings 396 and 397).

Listing 396: env-mlb11.yaml
modifyLineBreaks:
  environments:
    EndStartsOnOwnLine: 3

Listing 397: env-mlb12.yaml
modifyLineBreaks:
  environments:
    EndFinishesWithLineBreak: 3

Upon running commands analogous to the above, we obtain Listings 398 and 399.

Listing 398: env-mlb.tex using Listing 396
before words \begin{myenv}body of myenv
\end{myenv} after words

Listing 399: env-mlb.tex using Listing 397
before words \begin{myenv}body of myenv\end{myenv}
after words
Note that line breaks have been added as in Listings 390 and 391, and that a blank line has been added after the line break.

Let's now change each of the 1 values in Listings 396 and 397 so that they are 4 and save them into env-end4.yaml and env-end-f4.yaml respectively (see Listings 400 and 401).

![Listing 400: env-end4.yaml](image1)

![Listing 401: env-end-f4.yaml](image2)

We will demonstrate this poly-switch value using the code from Listing 385 on page 102.

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 402 and 403.

![Listing 402: env-mlb1.tex using Listing 400](image3)

![Listing 403: env-mlb1.tex using Listing 401](image4)

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

1. in Listing 402 a blank line has been inserted before the \end statement, even though the \end statement was already on its own line;
2. in Listing 403 a blank line has been inserted after the \end statement, even though it already began on its own line.

### 6.6.1.3 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 404 using poly-switch values of 1, 2, or 3, it will be left unchanged.

![Listing 404: env-mlb2.tex](image5)

![Listing 405: env-mlb3.tex](image6)

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 386 and 387 and Listings 402 and 403.

In contrast, the output from processing the file in Listing 405 will vary depending on the poly-switches used; in Listing 406 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 407 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 405 and by setting the other poly-switches considered so far to 2 in turn.
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 369 on page 101, an example is shown for the `equation*` environment.

### 6.6.1.4 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 408, noting in particular the positions of the line break highlighters, ♠, ♥, ♦ and ♣, together with the associated YAML files in Listings 409 to 412.

After running the commands

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

we obtain the respective output in Listings 413 to 416.
6.6 Poly-switches

Notice that in:

- Listing 413 the line break denoted by ♠ in Listing 408 has been removed;
- Listing 414 the line break denoted by ♥ in Listing 408 has been removed;
- Listing 415 the line break denoted by ♦ in Listing 408 has been removed;
- Listing 416 the line break denoted by ♣ in Listing 408 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 409 to 412 into one file; alternatively, you could tell latexindent.pl to load them all by using the following command, for example:

```
```

which gives the output in Listing 370 on page 101.

6.6.1.5 About trailing horizontal space

Recall that on page 30 we discussed the YAML field removeTrailingWhitespace, and that it has two (binary) switches to determine if horizontal space should be removed beforeProcessing and afterProcessing. The beforeProcessing is particularly relevant when considering the -m switch; let’s consider the file shown in Listing 417, which highlights trailing spaces.

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

```
\end{myenv}after words
```

The output from the following commands

```
```

```
cmh:~$ latexindent.pl -m env-mlb5.tex -l
env-mlb13.yaml,env-mlb14.yaml,env-mlb15.yaml,env-mlb16.yaml,removeTWS-before.yaml
```

is shown, respectively, in Listings 419 and 420; note that the trailing horizontal white space has been preserved (by default) in Listing 419, while in Listing 420, it has been removed using the switch specified in Listing 418.

```
before_words
\begin{myenv}
body_of_myenv\end{myenv}
```

```
\end{myenv}after_words
```

6.6.1.6 poly-switch line break removal and blank lines

Now let’s consider the file in Listing 421, which contains blank lines.
6.6 Poly-switches

We can explore this further using the blank-line poly-switch value of 3; let's use the file given in Listing 425.

Upon running the following commands

we receive the respective outputs given in Listings 426 and 427.
6.6 Poly-switches

Notice that in:

- Listing 426 that \end{one} has added a blank line, because of the value of EndFinishesWithLineBreak in Listing 397 on page 103, and even though the line break ahead of \begin{two} should have been removed (because of BeginStartsOnOwnLine in Listing 409 on page 105), the blank line has been preserved by default;
- Listing 427, by contrast, has had the additional line-break removed, because of the settings in Listing 422.

6.6.2 Poly-switches for double back slash

With reference to lookForAlignDelims (see Listing 33 on page 30) you can specify poly-switches to dictate the line-break behaviour of double back slashes in environments (Listing 35 on page 31), commands (Listing 69 on page 37), or special code blocks (Listing 104 on page 42). Note that for these poly-switches to take effect, the name of the code block must necessarily be specified within lookForAlignDelims (Listing 33 on page 30); we will demonstrate this in what follows.

Consider the code given in Listing 428.

Listing 428: tabular3.tex
\begin{tabular}{cc}
1 & 2 \\
\% 3 & 4 \\
\%
\end{tabular}

Referencing Listing 428:

- DBS stands for double back slash;
- line breaks ahead of the double back slash are annotated by ★, and are controlled by DBSStartsOnOwnLine;
- line breaks after the double back slash are annotated by □, and are controlled by DBSFinishesWithLineBreak.

Let's explore each of these in turn.

6.6.2.1 Double back slash starts on own line

We explore DBSStartsOnOwnLine (★ in Listing 428); starting with the code in Listing 428, together with the YAML files given in Listing 430 and Listing 432 and running the following commands

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

then we receive the respective output given in Listing 429 and Listing 431.

Listing 429: tabular3.tex using Listing 430
\begin{tabular}{cc}
1 & 2 \\
\% 3 & 4 \\
\%
\end{tabular}

Listing 430: DBS1.yaml
modifyLineBreaks:
environments:
DBSStartsOnOwnLine: 1

Listing 431: tabular3.tex using Listing 432
\begin{tabular}{cc}
1 & 2 \% 3 & 4 \%
\end{tabular}

Listing 432: DBS2.yaml
modifyLineBreaks:
environments:
tabular:
DBSStartsOnOwnLine: 2
We note that

- Listing 430 specifies \texttt{DBSStartsOnOwnLine} for every environment (that is within \texttt{lookForAlignDelims}, Listing 36 on page 31); the double back slashes from Listing 428 have been moved to their own line in Listing 429;

- Listing 432 specifies \texttt{DBSStartsOnOwnLine} on a per-name basis for tabular (that is within \texttt{lookForAlignDelims}, Listing 36 on page 31); the double back slashes from Listing 428 have been moved to their own line in Listing 431, having added comment symbols before moving them.

### 6.6.2.2 Double back slash finishes with line break

Let’s now explore \texttt{DBSFinishesWithLineBreak} (□ in Listing 428); starting with the code in Listing 428, together with the YAML files given in Listing 434 and Listing 436 and running the following commands

```bash
cmh:~$ latexindent.pl -m tabular3.tex -l DBS3.yaml
```
```
cmh:~$ latexindent.pl -m tabular3.tex -l DBS4.yaml
```

then we receive the respective output given in Listing 433 and Listing 435.

<table>
<thead>
<tr>
<th>Listing 433: tabular3.tex using Listing 434</th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{tabular}{cc} 1 &amp; 2 \ \ 3 &amp; 4 \ \ \ \end{tabular}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 434: DBS.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{modifyLineBreaks:}</td>
</tr>
<tr>
<td>\texttt{environments:}</td>
</tr>
<tr>
<td>\texttt{DBSFinishesWithLineBreak: 1}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 435: tabular3.tex using Listing 436</th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{tabular}{cc} 1 &amp; 2 % 3 &amp; 4 \ \ \ \end{tabular}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 436: DBS4.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{modifyLineBreaks:}</td>
</tr>
<tr>
<td>\texttt{environments:}</td>
</tr>
<tr>
<td>\texttt{tabular:}</td>
</tr>
<tr>
<td>\texttt{DBSFinishesWithLineBreak: 2}</td>
</tr>
</tbody>
</table>

We note that

- Listing 434 specifies \texttt{DBSFinishesWithLineBreak} for every environment (that is within \texttt{lookForAlignDelims}, Listing 36 on page 31); the code following the double back slashes from Listing 428 has been moved to their own line in Listing 433;

- Listing 436 specifies \texttt{DBSFinishesWithLineBreak} on a per-name basis for tabular (that is within \texttt{lookForAlignDelims}, Listing 36 on page 31); the double back slashes from Listing 428 have moved code following them to their own line in Listing 435, 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.

### 6.6.2.3 Double back slash poly-switches for \texttt{specialBeginEnd}

Let’s explore the double back slash poly-switches for code blocks within \texttt{specialBeginEnd} code blocks (Listing 102 on page 42); we begin with the code within Listing 437.

<table>
<thead>
<tr>
<th>Listing 437: special4.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{a &amp; =b \ \ \ &amp; =c\ \ &amp; =d\ \ &amp; =e}</td>
</tr>
</tbody>
</table>

Upon using the YAML settings in Listing 439, and running the command

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

then we receive the output given in Listing 438.
### 6.6 Poly-switches

#### 6.6.2.4 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 440.

**Listing 440:** mycommand2.tex

\[
\begin{array}{ccc}
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9 \\
10 & 11 & 12
\end{array}
\]

Upon using the YAML settings in Listings 442 and 444, and running the command

```
cmh:~$ latexindent.pl -m mycommand2.tex -l DBS6.yaml
```

then we receive the output given in Listings 441 and 443.

**Listing 441:** mycommand2.tex using Listing 442

\[
\begin{array}{ccc}
1 & 2 & 3 \\
\% \\
4 & 5 & 6 \\
7 & 8 & 9 \\
10 & 11 & 12
\end{array}
\]

There are a few things to note:

- in Listing 439 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.
6.6 Poly-switches

### 6.6.2.5 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 445

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

and running the following command, using Listing 434,

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

then we receive the output given in Listing 446.

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

You can customise the pattern for the double back slash by exploring the fine tuning field detailed in Listing 511 on page 132.

### 6.6.3 Poly-switches for other code blocks

Rather than repeat the examples shown for the environment code blocks (in Section 6.6.1 on page 100), we choose to detail the poly-switches for all other code blocks in Table 3; 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 70 on page 37) can not be modified using latexindent.pl. However, there are two poly-switches available for verbatim code blocks: environments (Listing 18 on page 27), commands (Listing 19 on page 27) and specialBeginEnd (Listing 115 on page 44).
### Table 3: 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>environment before words</td>
<td>♠ BeginStartsOnOwnLine</td>
<td></td>
</tr>
<tr>
<td>\begin{myenv}</td>
<td>♠ BodyStartsOnOwnLine</td>
<td></td>
</tr>
<tr>
<td>body of myenv</td>
<td>♠ EndStartsOnOwnLine</td>
<td></td>
</tr>
<tr>
<td>\end{myenv}</td>
<td>♠ EndFinishesWithLineBreak</td>
<td></td>
</tr>
<tr>
<td>ifelsefi</td>
<td>♠ IfStartsOnOwnLine</td>
<td></td>
</tr>
<tr>
<td>\if...\else\fi</td>
<td>♠ OrStartsOnOwnLine</td>
<td></td>
</tr>
<tr>
<td>body of if/or statement</td>
<td>♠ OrFinishesWithLineBreak</td>
<td></td>
</tr>
<tr>
<td>\or\else\fi</td>
<td>♠ ElseStartsOnOwnLine</td>
<td></td>
</tr>
<tr>
<td>body of else statement</td>
<td>♠ ElseFinishesWithLineBreak</td>
<td></td>
</tr>
<tr>
<td>\fi</td>
<td>♠ FiStartsOnOwnLine</td>
<td></td>
</tr>
<tr>
<td>optionalArguments</td>
<td>♠ LSqBStartsOnOwnLine(^8)</td>
<td></td>
</tr>
<tr>
<td>[</td>
<td>♠ OptArgBodyStartsOnOwnLine</td>
<td></td>
</tr>
<tr>
<td>value before comma</td>
<td>♠ CommaStartsOnOwnLine</td>
<td></td>
</tr>
<tr>
<td>]</td>
<td>♠ CommaFinishesWithLineBreak</td>
<td></td>
</tr>
<tr>
<td>mandatoryArguments</td>
<td>♠ LCuBStartsOnOwnLine(^9)</td>
<td></td>
</tr>
<tr>
<td>{</td>
<td>♠ MandArgBodyStartsOnOwnLine</td>
<td></td>
</tr>
<tr>
<td>value before comma</td>
<td>♠ CommaStartsOnOwnLine</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td>♠ CommaFinishesWithLineBreak</td>
<td></td>
</tr>
<tr>
<td>commands</td>
<td>♠ CommandStartsOnOwnLine</td>
<td></td>
</tr>
<tr>
<td>\mycommand (arguments)</td>
<td>♠ CommandNameFinishesWithLineBreak</td>
<td></td>
</tr>
<tr>
<td>namedGroupingBraces Brackets</td>
<td>♠ NameStartsOnOwnLine</td>
<td></td>
</tr>
<tr>
<td>myname (braces/brackets)</td>
<td>♠ NameFinishesWithLineBreak</td>
<td></td>
</tr>
<tr>
<td>keyEqualsValuesBracesBrackets</td>
<td>♠ KeyStartsOnOwnLine</td>
<td></td>
</tr>
<tr>
<td>key = (braces/brackets)</td>
<td>♠ EqualsStartsOnOwnLine</td>
<td></td>
</tr>
<tr>
<td>items</td>
<td>♠ ItemStartsOnOwnLine</td>
<td></td>
</tr>
<tr>
<td>\item</td>
<td>♠ ItemFinishesWithLineBreak</td>
<td></td>
</tr>
<tr>
<td>specialBeginEnd</td>
<td>♠ SpecialBeginStartsOnOwnLine</td>
<td></td>
</tr>
<tr>
<td>[</td>
<td>♠ SpecialBodyStartsOnOwnLine</td>
<td></td>
</tr>
<tr>
<td>body of special/middle</td>
<td>♠ SpecialMiddleStartsOnOwnLine</td>
<td></td>
</tr>
<tr>
<td>\middle</td>
<td>♠ SpecialMiddleFinishesWithLineBreak</td>
<td></td>
</tr>
<tr>
<td>body of special/middle</td>
<td>♠ SpecialEndStartsOnOwnLine</td>
<td></td>
</tr>
<tr>
<td>]</td>
<td>♠ SpecialEndFinishesWithLineBreak</td>
<td></td>
</tr>
<tr>
<td>verbatim</td>
<td>♠ VerbatimBeginStartsOnOwnLine</td>
<td></td>
</tr>
</tbody>
</table>

\(^8\) LSqB stands for Left Square Bracket  
\(^9\) LCuB stands for Left Curly Brace
6.6 Poly-switches

6.6.4 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 3 on the preceding page) and LCuBStartsOnOwnLine for mandatory arguments, and LSqBStartsOnOwnLine for optional arguments.

Let’s begin with the code in Listing 447 and the YAML settings in Listing 449; with reference to Table 3 on the previous page, the key CommandNameFinishesWithLineBreak is an alias for BodyStartsOnOwnLine.

\begin{verbatim}
mycommand
{  
  mand arg text  
  mand arg text}
{  
  mand arg text  
  mand arg text}
\end{verbatim}

LISTING 447: mycommand1.tex

Upon running the command

\begin{verbatim}
\$ latexindent.pl -m -l=mycom-mlb1.yaml mycommand1.tex
\end{verbatim}

we obtain Listing 448; note that the second mandatory argument beginning brace \} has had its leading line break removed, but that the first brace has not.

\begin{verbatim}
/\mycommand
{  
  mand arg text  
  mand arg text}\{  
  mand arg text  
  mand arg text}
\end{verbatim}

LISTING 448: mycommand1.tex using Listing 449

\begin{verbatim}
modifyLineBreaks:  
  commands:  
    CommandNameFinishesWithLineBreak: 0  
    mandatoryArguments:  
      LCuBStartsOnOwnLine: -1
\end{verbatim}

LISTING 449: mycom-mlb1.yaml

Now let’s change the YAML file so that it is as in Listing 451; upon running the analogous command to that given above, we obtain Listing 450; both beginning braces \} have had their leading line breaks removed.

\begin{verbatim}
\mycommand{  
  mand arg text  
  mand arg text}\{  
  mand arg text  
  mand arg text}
\end{verbatim}

LISTING 450: mycommand1.tex using Listing 451

\begin{verbatim}
modifyLineBreaks:  
  commands:  
    CommandNameFinishesWithLineBreak: -1  
    mandatoryArguments:  
      LCuBStartsOnOwnLine: -1
\end{verbatim}

LISTING 451: mycom-mlb2.yaml

Now let’s change the YAML file so that it is as in Listing 453; upon running the analogous command to that given above, we obtain Listing 452.
### 6.6 Poly-switches

#### Listing 452: mycommand1.tex using Listing 453

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

#### Listing 453: mycom-mlb3.yaml

```yaml
modifyLineBreaks:
  commands:
    CommandNameFinishesWithLineBreak: -1
  mandatoryArguments:
    LCuBStartsOnOwnLine: 1
```

#### Listing 454: mycommand1.tex using Listing 455

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

#### Listing 455: mycom-mlb4.yaml

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

### 6.6.5 Conflicting poly-switches: sequential code blocks

It is very easy to have conflicting poly-switches; if we use the example from Listing 447 on the preceding page, and consider the YAML settings given in Listing 455. The output from running `cmh:~$ latexindent.pl -m -l=mycom-mlb4.yaml mycommand1.tex` is given in Listing 455.

#### Listing 456: mycommand1.tex using Listing 457

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

#### Listing 457: mycom-mlb5.yaml

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

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 459, which give associated output in Listing 458.
6.6 Poly-switches

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.

6.6.6 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 460, noting that it contains nested environments.

Let’s use the YAML settings given in Listing 462, which upon running the command

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

gives the output in Listing 461.

In Listing 461, 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, and then sequentially – for example, in Listing 460, 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);
6.6 Poly-switches

- 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 -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 \textit{NOT} added (see \texttt{EndFinishesWithLineBreak}).

With reference to Listing 461, 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{EndStartsOnOwnLine} is set to \texttt{-1}. Importantly, because, \textit{at this stage}, \texttt{\end{two}} \textit{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 464; upon running the command

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

we obtain the output given in Listing 463.

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

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}} \textit{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 \texttt{-r}, \texttt{-rv} and \texttt{-rr} switches

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

- the \texttt{-r} switch will perform indentation and replacements, not respecting verbatim code blocks;
- the \texttt{-rv} switch will perform indentation and replacements, and \textit{will} respect verbatim code blocks;
- the \texttt{-rr} switch will \textit{not} perform indentation, and will perform replacements not respecting verbatim code blocks.

We will demonstrate each of the \texttt{-r}, \texttt{-rv} and \texttt{-rr} switches, but a summary is given in Table 4.

<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 \texttt{replacements} field is shown in Listing 465; as with all of the other fields, you are encouraged to customise and change this as you see fit. The options in this field will \textit{only} be considered if the \texttt{-r}, \texttt{-rv} or \texttt{-rr} switches are active; when discussing YAML settings related to the replacement-mode switches, we will use the style given in Listing 465.

\begin{Verbatim}[commandchars=\][\]]
replacements:
- amalgamate: 1
- this: 'latexindent.pl'
  that: 'pl.latexindent'
lookForThis: 1
when: before
\end{Verbatim}

The first entry within the \texttt{replacements} field is \texttt{amalgamate}, and is \textit{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 \textit{one} entry in the \texttt{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 465, the default action will replace every instance of the text \texttt{latexindent.pl} with \texttt{pl.latexindent}.

Beginning with the code in Listing 466 and running the command

\begin{Verbatim}[commandchars=\][\]]
\texttt{cmh:~}\$ \texttt{latexindent.pl -r replace1.tex}
\end{Verbatim}
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.

### Example 1

We begin with code given in Listing 470

```latex
\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 472, and running the command

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

then we achieve the output in Listing 471.
7.3 Examples of replacements

Note that in Listing 472, 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 472 more concise by exploring the substitution field. Using the settings in Listing 474 and running the command

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

then we achieve the output in Listing 473.

The code given in Listing 474 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, [10] for a detailed covering of the topic. With reference to Listing 474, 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 sg at the end of the line) instructs latexindent.pl to treat your file as one single line;
- the g modifier (in the 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 474 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 sg modifiers for this feature.

Example 2

We’ll keep working with the file in Listing 470 on the preceding page for this example.

Using the YAML in Listing 476, and running the command

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

then we achieve the output in Listing 475.
### Listing 475: `colsep.tex` using `multi-line!`

With reference to Listing 476, 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 465 on page 117. 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 478, and running the command

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

then we achieve the output in Listing 477.

### Listing 477: `colsep.tex` using `multi-line1.yaml`

We note that, because we have specified `when: after`, that `latexindent.pl` has not found the string specified in Listing 478 within the file in Listing 470 on page 118. As it has looked for the string within Listing 478 after the indentation has been performed. After indentation, the string as written in Listing 478 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 475.

### Example 3

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

Assuming that we start with the code in Listing 479, 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 479:** 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 481 and run the command

```bash
cmh:~$ latexindent.pl -r displaymath.tex -l=displaymath1.yaml
```

to receive the output given in Listing 480.

**LISTING 480:** displaymath.tex using Listing 481

```latex
\begin{equation*}
a^2+b^2=4\end{equation*}
\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: $h^2$
```

**LISTING 481:** displaymath1.yaml

replacements:

- substitution: |-
  s/$\$
  (.*)
  \$/\begin{equation*}$1\end{equation*}/sgx

A few notes about Listing 481:

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 6.6 on page 100, which we do in Listing 483; upon running the command

```bash
cmh:~$ latexindent.pl -r -m displaymath.tex -l=displaymath1.yaml,equation.yaml
```

then we receive the output in Listing 482.
7.3 Examples of replacements

Example 4
This example is motivated by tex stackexchange question 490086. We begin with the code in Listing 484.

Our goal is to make the spacing uniform between the phrases. To achieve this, we employ the settings in Listing 486, and run the command

```
\texttt{cmsh:~\$ latexindent.pl \textasciitilde r phrase.tex \textasciitilde l=hspace.yaml}
```

which gives the output in Listing 485.

The \h+ setting in Listing 486 say to replace at least one horizontal space with a single space.
7.3 Examples of replacements

Example 5  We begin with the code in Listing 487.

**LISTING 487: references.tex**

```latex
\text{equation \eqref{eq:aa} and Figure \ref{fig:bb} and table~\ref{tab:cc}}
```

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 489 and running the command

```bash
$ latexindent -r references.tex -l=reference.yaml
```

which gives the output in Listing 488.

**LISTING 488: references.tex using Listing 489**

```latex
\text{\hyperref{equation \ref*{eq:aa}} and \hyperref{Figure \ref*{fig:bb}} and \hyperref{table \ref*{tab:cc}}}
```

**LISTING 489: reference.yaml**

```
replacements:
  - substitution: |
    s/(equation | table | figure | section )\(?:eq)?\ref\{(.*?)\}/\hyperref{$1 \ref\*{$3}}/sgxi
```

Referencing Listing 489, 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 4 on page 117) in the context of an example that contains a verbatim code block, Listing 490; we will use the settings in Listing 491.

**LISTING 490: verbi.tex**

```
\begin{myenv}
body of verbatim
\end{myenv}
some verbatim
\begin{verbatim}
body
of
verbatim
text
\end{verbatim}
text
```

Upon running the following commands,
7.3 Examples of replacements

We receive the respective output in Listings 492 to 494.

<table>
<thead>
<tr>
<th>Listing 492: verb1-mod1.tex</th>
<th>Listing 493: verb1-rv-mod1.tex</th>
<th>Listing 494: verb1-rr-mod1.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin{myenv}</td>
<td>\begin{myenv}</td>
<td>\begin{myenv}</td>
</tr>
<tr>
<td>head of verbatim</td>
<td>head of verbatim</td>
<td>head of verbatim</td>
</tr>
<tr>
<td>\end{myenv}</td>
<td>\end{myenv}</td>
<td>\end{myenv}</td>
</tr>
<tr>
<td>some verbatim</td>
<td>some verbatim</td>
<td>some verbatim</td>
</tr>
<tr>
<td>\begin{verbatim}</td>
<td>\begin{verbatim}</td>
<td>\begin{verbatim}</td>
</tr>
<tr>
<td>head</td>
<td>body</td>
<td>head</td>
</tr>
<tr>
<td>of</td>
<td>of</td>
<td>of</td>
</tr>
<tr>
<td>verbatim</td>
<td>verbatim text</td>
<td>verbatim text</td>
</tr>
<tr>
<td>text</td>
<td>text</td>
<td>text</td>
</tr>
<tr>
<td>\end{verbatim}</td>
<td>\end{verbatim}</td>
<td>\end{verbatim}</td>
</tr>
<tr>
<td>text</td>
<td>text</td>
<td>text</td>
</tr>
</tbody>
</table>

We note that:

1. in Listing 492 indentation has been performed, and that the replacements specified in Listing 491 have been performed, even within the verbatim code block;
2. in Listing 493 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 494 indentation has not been performed, but that replacements have been performed, not respecting the verbatim code block.

See the summary within Table 4 on page 117.

\textbf{Example 7}

Let’s explore the \texttt{amalgamate} field from Listing 465 on page 117 in the context of the file specified in Listing 495.

<table>
<thead>
<tr>
<th>Listing 495: amalg1.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>one two three</td>
</tr>
</tbody>
</table>

Let’s consider the YAML files given in Listings 496 to 498.

<table>
<thead>
<tr>
<th>Listing 496: amalg1-yaml.yaml</th>
<th>Listing 497: amalg2-yaml.yaml</th>
<th>Listing 498: amalg3-yaml.yaml</th>
</tr>
</thead>
<tbody>
<tr>
<td>replacements:</td>
<td>replacements:</td>
<td>replacements:</td>
</tr>
<tr>
<td>this: one</td>
<td>this: two</td>
<td>this: three</td>
</tr>
<tr>
<td>that: 1</td>
<td>that: 2</td>
<td>that: 3</td>
</tr>
</tbody>
</table>

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 499 to 501.
### 7.3 Examples of replacements

<table>
<thead>
<tr>
<th>LISTING 499: amalg1.tex using Listing 496</th>
<th>LISTING 500: amalg1.tex using Listings 496 and 497</th>
<th>LISTING 501: amalg1.tex using Listings 496 to 498</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 499 the replacements from Listing 496 have been used;
2. in Listing 500 the replacements from Listings 496 and 497 have both been used, because the default value of \texttt{amalgamate} is 1;
3. in Listing 501 only the replacements from Listing 498 have been used, because the value of \texttt{amalgamate} has been set to 0.
SECTION 8

The –lines switch

latexindent.pl can operate on a selection of lines of the file using the –lines or –n switch. The basic syntax is –lines MIN-MAX, so for example

```
cmh:~$ latexindent.pl --lines 3-7 myfile.tex
```
```
cmh:~$ latexindent.pl -n 3-7 myfile.tex
```

will only operate upon lines 3 to 7 in myfile.tex. All of the other lines will not be operated upon by latexindent.pl.

The options for the lines switch are:

- line range, as in –lines 3-7
- single line, as in –lines 5
- multiple line ranges separated by commas, as in –lines 3-5,8-10
- negated line ranges, as in –lines !3-5 which translates to –lines 1-2,6-N, where N is the number of lines in your file.

We demonstrate this feature, and the available variations in what follows. We will use the file in Listing 502.

**Listing 502: myfile.tex**

```latex
1 Before the environments
2 \begin{one}
3 first block, first line
4 first block, second line
5 first block, third line
6 \begin{two}
7 second block, first line
8 second block, second line
9 second block, third line
10 second block, fourth line
11 \end{two}
12 \end{one}
```

**Example 8** We demonstrate the basic usage using the command

```
cmh:~$ latexindent.pl --lines 3-7 myfile.tex -o=+-mod1
```

which instructs latexindent.pl to only operate on lines 3 to 7; the output is given in Listing 503.
### Example 9
You can call the `lines` switch with only *one number* and in which case only that line will be operated upon. For example

```
cmh:~$ latexindent.pl --lines 5 myfile.tex -o=+-mod2
```

instructs `latexindent.pl` to only operate on line 5; the output is given in Listing 504.

<table>
<thead>
<tr>
<th>Listing 504: myfile-mod2.tex</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Before the environments</td>
</tr>
<tr>
<td>2 \begin{one}</td>
</tr>
<tr>
<td>3 first block, first line</td>
</tr>
<tr>
<td>4 first block, second line</td>
</tr>
<tr>
<td>5 first block, third line</td>
</tr>
<tr>
<td>6 \begin{two}</td>
</tr>
<tr>
<td>7 second block, first line</td>
</tr>
<tr>
<td>8 second block, second line</td>
</tr>
<tr>
<td>9 second block, third line</td>
</tr>
<tr>
<td>10 second block, fourth line</td>
</tr>
<tr>
<td>11 \end{two}</td>
</tr>
<tr>
<td>12 \end{one}</td>
</tr>
</tbody>
</table>

The following two calls are equivalent:

```
cmh:~$ latexindent.pl --lines 5 myfile.tex
```
```
cmh:~$ latexindent.pl --lines 5-5 myfile.tex
```

### Example 10
If you specify a value outside of the line range of the file then `latexindent.pl` will ignore the `lines` argument, detail as such in the log file, and proceed to operate on the entire file.

For example, in the following call

```
cmh:~$ latexindent.pl --lines 11-13 myfile.tex
```
latexindent.pl will ignore the lines argument, and operate on the entire file because Listing 502 only has 12 lines.

Similarly, in the call

```
cmh:~$ latexindent.pl --lines -1-3 myfile.tex
```

latexindent.pl will ignore the lines argument, and operate on the entire file because we assume that negatively numbered lines in a file do not exist.

**Example 11** You can specify multiple line ranges as in the following

```
cmh:~$ latexindent.pl --lines 3-5,8-10 myfile.tex -o=+-mod3
```

which instructs latexindent.pl to operate upon lines 3 to 5 and lines 8 to 10; the output is given in Listing 505.

```
\begin{one}
first block, first line
first block, second line
first block, third line
  \begin{two}
    second block, first line
    second block, second line
    second block, third line
  \end{two}
second block, fourth line
\end{one}
```

The following calls to latexindent.pl are all equivalent

```
cmh:~$ latexindent.pl --lines 3-5,8-10 myfile.tex
```

as latexindent.pl performs a check to put the lowest line ranges first, and within each line range, it puts the lowest number first.

**Example 12** There’s no limit to the number of line ranges that you can specify, they just need to be separated by commas. For example

```
cmh:~$ latexindent.pl --lines 1-2,4-5,9-10,12 myfile.tex -o=+-mod4
```

has four line ranges: lines 1 to 2, lines 4 to 5, lines 9 to 10 and line 12. The output is given in Listing 506.
Listing 506: myfile-mod4.tex

1  Before the environments
2  \begin{one}
3     first block, first line
4     first block, second line
5     first block, third line
6  \begin{two}
7     second block, first line
8     second block, second line
9     second block, third line
10    second block, fourth line
11  \end{two}
12  \end{one}

As previously, the ordering does not matter, and the following calls to `latexindent.pl` are all equivalent:

\begin{verbatim}
cmh:~$ latexindent.pl --lines 1-2,4-5,9-10,12 myfile.tex
cmh:~$ latexindent.pl --lines 2-1,4-5,9-10,12 myfile.tex
cmh:~$ latexindent.pl --lines 4-5,1-2,9-10,12 myfile.tex
cmh:~$ latexindent.pl --lines 12,4-5,1-2,9-10 myfile.tex
\end{verbatim}

as `latexindent.pl` performs a check to put the lowest line ranges first, and within each line range, it puts the lowest number first.

Example 13 You can specify negated line ranges by using `!` as in

\begin{verbatim}
cmh:~$ latexindent.pl --lines !5-7 myfile.tex -o=+-mod5
\end{verbatim}

which instructs `latexindent.pl` to operate upon all of the lines except lines 5 to 7.

In other words, `latexindent.pl` will operate on lines 1 to 4, and 8 to 12, so the following two calls are equivalent:

\begin{verbatim}
cmh:~$ latexindent.pl --lines !5-7 myfile.tex
cmh:~$ latexindent.pl --lines 1-4,8-12 myfile.tex
\end{verbatim}

The output is given in Listing 507.

Listing 507: myfile-mod5.tex

1  Before the environments
2  \begin{one}
3     first block, first line
4     first block, second line
5     first block, third line
6  \begin{two}
7     second block, first line
8     second block, second line
9     second block, third line
10    second block, fourth line
11  \end{two}
12  \end{one}
Example 14  You can specify *multiple negated line ranges* such as

```
cmh:∼$ latexindent.pl --lines !5-7,!9-10 myfile.tex -o=+-mod6
```

which is equivalent to:

```
cmh:∼$ latexindent.pl --lines 1-4,8,11-12 myfile.tex -o=+-mod6
```

The output is given in Listing 508.

```
1 Before the environments
2 \begin{one}
3 first block, first line
4 first block, second line
5 first block, third line
6 \begin{two}
7 second block, first line
8 second block, second line
9 second block, third line
10 second block, fourth line
11 \end{two}
12 \end{one}
```

Example 15  If you specify a line range with anything other than an integer, then latexindent.pl will ignore the `lines` argument, and *operate on the entire file*.

Sample calls that result in the `lines` argument being ignored include the following:

```
cmh:∼$ latexindent.pl --lines 1-x myfile.tex

```

```
cmh:∼$ latexindent.pl --lines !y-3 myfile.tex
```

Example 16  We can, of course, use the `lines` switch in combination with other switches.

For example, let's use with the file in Listing 509.

```
1 Before the environments
2 \begin{one}
3 first block, first line
4 first block, second line
5 first block, third line
6 \begin{two} body \end{two}
7 \end{one}
```

We can demonstrate interaction with the `-m` switch (see Section 6 on page 71); in particular, if we use Listing 404 on page 104, Listing 388 on page 103 and Listing 389 on page 103 and run

```
cmh:∼$ latexindent.pl --lines 6 myfile1.tex -o=+=mod1 -m -l env-mlb2,env-mlb7,env-mlb8 -o=+=mod1
```

then we receive the output in Listing 510.
Before the environments
\begin{one}
  first block, first line
  first block, second line
  first block, third line
\end{one}
\begin{two}
  body
\end{two}
\end{one}
SECTION 9

Fine tuning

latexindent.pl operates by looking for the code blocks detailed in Table 2 on page 49. The fine tuning of the details of such code blocks is controlled by the fineTuning field, detailed in Listing 511. 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 511 are all regular expressions. This manual is not intended to be a tutorial on regular expressions; you might like to read, for example, [10] for a detailed covering of the topic.

We make the following comments with reference to Listing 511:

1. the environments:name field details that the name of an environment can contain:

   (a) a-z lower case letters
   (b) A-Z upper case letters
   (c) @ the @ 'letter'
   (d) * stars
   (e) 0-9 numbers
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 \ 

4. in the arguments:before field
   (a) \d+\* 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 71. In particular:
   (a) betterFullStop is in relation to the one sentence per line routine, detailed in Section 6.5 on page 91
   (b) doubleBackSlash is in relation to the DBSStartsOnOwnLine and DBSFinishesWithLineBreak polyswitches surrounding double backslashes, see Section 6.6.2 on page 108
   (c) comma is in relation to the CommaStartsOnOwnLine and CommaFinishesWithLineBreak polyswitches surrounding commas in optional and mandatory arguments; see Table 3 on page 112

It is not obvious from Listing 511, but each of the follow, before and between fields allow trailing comments, line breaks, and horizontal spaces between each character.

Example 17 As a demonstration, consider the file given in Listing 512, together with its default output using the command

```bash
$ latexindent.pl finetuning1.tex
```

is given in Listing 513.

<table>
<thead>
<tr>
<th>Listing 512: finetuning1.tex</th>
<th>Listing 513: 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 513 that the indentation scheme has not worked as expected. We can
fine tune the indentation scheme by employing the settings given in Listing 515 and running the command

```
cmh:~$ latexindent.pl finetuning1.tex -l=fine-tuning1.yaml
```

and the associated (desired) output is given in Listing 514.

**Listing 514:** finetuning1.tex using Listing 515

```latex
\mycommand{
  \rule{G -> +H[-G]CL}
  \rule{H -> -G[+H]CL}
  \rule{g -> +h[-g]cL}
  \rule{h -> -g[+h]cL}
}
```

**Listing 515:** finetuning1.yaml

```
fineTuning:
  arguments:
    between:
      '-_\^*|->|->|\+[h|H|g|G'
```

**Example 18** Let's have another demonstration; consider the file given in Listing 516, together with its default output using the command

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

is given in Listing 517.

**Listing 516:** finetuning2.tex

```latex
@misc{ wikilatex,  
author = "{Wikipedia contributors}",  
title = "LaTeX --- {Wikipedia}{,}",  
note = "[Online; accessed 3-March-2020]"
}
```

**Listing 517:** finetuning2.tex default

```latex
@misc{ wikilatex,  
author = "{Wikipedia contributors}",  
title = "LaTeX --- {Wikipedia}{,}",  
note = "[Online; accessed 3-March-2020]"
}
```

It's clear from Listing 517 that the indentation scheme has not worked as expected. We can fine tune the indentation scheme by employing the settings given in Listing 519 and running the command

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

and the associated (desired) output is given in Listing 518.

**Listing 518:** finetuning2.tex using Listing 519

```latex
@misc{ wikilatex,  
author = "{Wikipedia contributors}",  
title = "LaTeX --- {Wikipedia}{,}",  
note = "[Online; accessed 3-March-2020]"
}
```

**Listing 519:** finetuning2.yaml

```
fineTuning:
  NamedGroupingBracesBrackets:
    follow: '\\R\\{\{\{\$\}\}\}\'{
  UnNamedGroupingBracesBrackets:
    follow: '\\{\{\{\$\}\}\}\'{
  arguments:
    between: '_\^*|->|->|\+[h|H|g|G'
```

In particular, note that the settings in Listing 519 specify that NamedGroupingBracesBrackets and UnNamedGroupingBracesBrackets can follow " and that we allow --- between arguments.

**Example 19** You can tweak the fineTuning using the -y switch, but to be sure to use quotes appropriately. For example, starting with the code in Listing 520 and running the following command

```
cmh:~$ latexindent.pl finetuning2.tex -y
```
Example 20

We can tweak the fineTuning for how trailing comments are classified. For motivation, let’s consider the code given in Listing 522.

```latex
\begin{lstlisting}
some before text
\href{Handbook\%20for\%30Spoken\%40document.pdf}{my document}
some after text
\end{lstlisting}
```

We will compare the settings given in Listings 523 and 524.

```yaml
\begin{lstlisting}
modifyLineBreaks:
  textWrapOptions:
    columns: 80
    all: 1
    perCodeBlockBasis: 1
removeParagraphLineBreaks:
  all: 1
\end{lstlisting}
```

Upon running the following commands

```bash
cmh:~$ latexindent.pl -m finetuning4.tex -o=+-mod1 -l=href1
cmh:~$ latexindent.pl -m finetuning4.tex -o=+-mod2 -l=href2
```

we receive the respective output in Listings 525 and 526.

```latex
\begin{lstlisting}
some before text \href{Handbook\%20for\%30Spoken\%40document.pdf}{my document}
some after text
\end{lstlisting}
```
• Listing 525 the trailing comments are assumed to be everything following the first comment symbol, which has meant that everything following it has been moved to the end of the line; this is undesirable, clearly!

• Listing 526 has fine-tuned the trailing comment matching, and says that % cannot be immediately preceeded by the words ‘Handbook’, ‘for’ or ‘Spoken’, which means that none of the % symbols have been treated as trailing comments, and the output is desirable.

Another approach to this situation, which does not use fineTuning, is to use noIndentBlock which we discussed in Listing 20 on page 27; using the settings in Listing 527 and running the command

```
cmh:~$ latexindent.pl -m finetuning4.tex -o=-mod3 -l=href3
```

then we receive the same output given in Listing 526; see also paragraphsStopAt in Listing 309 on page 87.

```
modifyLineBreaks:
  textWrapOptions:
    columns: 80
    all: 1
    perCodeBlockBasis: 1
  removeParagraphLineBreaks:
    all: 1
    paragraphsStopAt:
      verbatim: 0

noIndentBlock:
  href:
    begin: '\href\{[^}\]*?\}\{'
    body: '[^\}]*?'
    end: '\}'
```

With reference to the body field in Listing 527, we note that the body field can be interpreted as: the fewest number of zero or more characters that are not right braces. This is an example of character class.
SECTION 10

Conclusions and known limitations

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

For example, with reference to the multicolumn alignment routine in Listing 46 on page 33, when working with code blocks in which multicolumn commands overlap, the algorithm can fail.

Another limitation is to do with efficiency, particularly when the \texttt{-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 115); I hope 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 16 on page 25) will be consulted. If you find a case in which the script struggles, please feel free to report it at [11], and in the meantime, consider using a \texttt{noIndentBlock} (see page 27).

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


## Section 11

### References

11.1 External links


[21] *perldoc Encode::Supported.* URL: https://perldoc.perl.org/Encode::Supported (visited on 05/06/2021).


[27] *Video demonstration of latexindent.pl on youtube.* URL: https://www.youtube.com/watch?v=wo38aahH2F4E&spfreload=10 (visited on 02/21/2017).


11.2 Contributors


11.2 Contributors


SECTION A

Required Perl modules

If you intend to use \texttt{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 528 (\texttt{perl helloworld.pl}) then you will be able to run \texttt{latexindent.pl}, otherwise you may need to install the missing modules – see appendices A.1 and A.2.

\begin{Verbatim}
#!/usr/bin/perl

use strict;
use warnings;
use utf8;
use PerlIO::encoding;
use Unicode::GString;
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 Encode;
use Getopt::Long;
use Data::Dumper;
use List::Util qw(max);

print "hello\_world"
exit;
\end{Verbatim}

A.1 Module installer script

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

\begin{Verbatim}
 cmb:~$ perl latexindent-module-installer.pl
\end{Verbatim}

or

\begin{Verbatim}
 C:\Users\cmb> perl latexindent-module-installer.pl
\end{Verbatim}

then, once you have answered Y, the appropriate modules will be installed onto your distribution.

A.2 Manually installing modules

Manually installing the modules given in Listing 528 will vary depending on your operating system and Perl distribution.
A.2 Manually installing modules

A.2.1 Linux

A.2.1.1 perlbrew

Linux users may be interested in exploring Perlbrew [20]; an example installation would be:

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

A.2.1.2 Ubuntu/Debian

For other distributions, the Ubuntu/Debian approach may work as follows:

```bash
cmh:~$ sudo apt install perl
```

or else by running, for example,

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

A.2.1.3 Ubuntu: using the texlive from apt-get

Ubuntu users that install texlive using apt-get as in the following

```bash
cmh:~$ sudo apt install texlive
```

may need the following additional command to work with latexindent.pl

```bash
cmh:~$ sudo apt install texlive-extra-utils
```

A.2.1.4 Alpine

If you are using Alpine, some Perl modules are not build-compatible with Alpine, but replacements are available through apk. For example, you might use the commands given in Listing 529; thanks to [12] for providing these details.
A.2 Manually installing modules

LISTING 529: alpine-install.sh

```bash
# Installing perl
apk --no-cache add miniperl perl-utils

# Installing incompatible latexindent perl dependencies via apk
apk --no-cache add \\n  perl-log-dispatch \\n  perl-namespace-autoclean \\n  perl-specio \\n  perl-unicode-linebreak

# Installing remaining latexindent perl dependencies via cpan
apk --no-cache add curl wget make
ls /usr/share/texmf-dist/scripts/latexindent
cd /usr/local/bin && \\
  curl -L https://cpanmin.us/ -o cpanm && \\
  chmod +x cpanm
cpanm -n App::cpanminus
cpanm -n File::HomeDir
cpanm -n Params::ValidationCompiler
cpanm -n YAML::Tiny
cpanm -n Unicode::GCString
```

Users of NixOS might like to see https://github.com/cmhughes/latexindent.pl/issues/222 for tips.

A.2.2 Mac

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

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

A.2.3 Windows

Strawberry Perl users on Windows might use CPAN client. All of the modules are readily available on CPAN [5].

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
```
latexindent.pl has a few scripts (available at [11]) that can update the path variables. Thank you to [14] for this feature. If you’re on a Linux or Mac machine, then you’ll want CMakeLists.txt from [11].

B.1 Add to path for Linux

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

1. download latexindent.pl and its associated modules, defaultSettings.yaml, to your chosen directory from [11];
2. within your directory, create a directory called path-helper-files and download CMakeLists.txt and cmake_uninstall.cmake.in from [11]/path-helper-files to this directory;
3. run

```
cmh:~$ ls /usr/local/bin
```
to see what is currently in there;
4. run the following commands

```
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
```
5. run

```
cmh:~$ ls /usr/local/bin
```
again to check that latexindent.pl, its modules and defaultSettings.yaml have been added.

To remove the files, run

```
cmh:~$ sudo make uninstall
```

B.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 [11] to your chosen directory;
2. open a command prompt and run the following command 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

```
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.
Listing 17 on page 26 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 530, and the settings specified in Listing 531.

**Listing 530: simple.tex**
\begin{myenv}
body of myenv
\end{myenv}

**Listing 531: logfile-prefs1.yaml**

```yaml
logFilePreferences:
  showDecorationStartCodeBlockTrace: "+++++
  showDecorationFinishCodeBlockTrace: "-----"
```

If we run the following command (noting that -t is active)

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

then on inspection of indent.log we will find the snippet given in Listing 532.

**Listing 532: indent.log**

```
+++++
TRACE: environment found: myenv
No ancestors found for myenv
Storing settings for myenv
indentRulesGlobal specified (0) for environments, ...
Using defaultIndent for myenv
Putting linebreak after replacementText for myenv
looking for COMMANDS and key = {value}
TRACE: Searching for commands with optional and/or mandatory arguments AND key = {value}
    looking for SPECIAL begin/end
TRACE: Searching myenv for special begin/end (see specialBeginEnd)
TRACE: Searching myenv for optional and mandatory arguments
    ... no arguments found
-----
```

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

Encoding indentconfig.yaml

In relation to Section 4 on page 21, Windows users that encounter encoding issues with `indentconfig.yaml`, may wish to run the following command in either `cmd.exe` or `powershell.exe`:

```
C:\Users\cmh> chcp
```

They may receive the following result

```
C:\Users\cmh> Active code page: 936
```

and can then use the settings given in Listing 533 within their `indentconfig.yaml`, where 936 is the result of the `chcp` command.

```
Listing 533: encoding demonstration for indentconfig.yaml

encoding: cp936
```
SECTION E

dos2unix linebreak adjustment

dos2unixlinebreaks: \{integer\}

If you use latexindent.pl on a dos-based Windows file on Linux then you may find that trailing horizontal space is not removed as you hope.

In such a case, you may wish to try setting dos2unixlinebreaks to 1 and employing, for example, the following command.

```cmh:
$ latexindent.pl -y="dos2unixlinebreaks:1" myfile.tex
```

See [29] for further details.
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,

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

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

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

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

The fields given in Listing 534 are obsolete from Version 3.0 onwards.

```
<table>
<thead>
<tr>
<th>Listing 534: Obsolete YAML fields from Version 3.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>alwaysLookforSplitBrackets</td>
</tr>
<tr>
<td>alwaysLookforSplitBracketsElse</td>
</tr>
<tr>
<td>checkunmatched</td>
</tr>
<tr>
<td>checkunmatchedELSE</td>
</tr>
<tr>
<td>checkunmatchedbracket</td>
</tr>
<tr>
<td>constructIfElseFi</td>
</tr>
</tbody>
</table>
```

There is a slight difference when specifying indentation after headings; specifically, we now write `indentAfterThisHeading` instead of `indent`. See Listings 535 and 536

```
<table>
<thead>
<tr>
<th>Listing 535: indentAfterThisHeading in Version 2.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>indentAfterHeadings:</td>
</tr>
<tr>
<td>part:</td>
</tr>
<tr>
<td>indent: 0</td>
</tr>
<tr>
<td>level: 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 536: indentAfterThisHeading in Version 3.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>indentAfterHeadings:</td>
</tr>
<tr>
<td>part:</td>
</tr>
<tr>
<td>indentAfterThisHeading: 0</td>
</tr>
<tr>
<td>level: 1</td>
</tr>
</tbody>
</table>
```

To specify noAdditionalIndent for display-math environments in Version 2.2, you would write YAML as in Listing 537; as of Version 3.0, you would write YAML as in Listing 538 or, if you’re using -m switch, Listing 539.
<table>
<thead>
<tr>
<th>Listing 537: noAdditionalIndent in Version 2.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>`noAdditionalIndent:</td>
</tr>
<tr>
<td>\[:, 0</td>
</tr>
<tr>
<td>\]: 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 538: noAdditionalIndent for displayMath in Version 3.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>specialBeginEnd:</td>
</tr>
<tr>
<td>displayMath:</td>
</tr>
<tr>
<td>begin: '```'</td>
</tr>
<tr>
<td>end: '```'</td>
</tr>
<tr>
<td>lookForThis: 0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Listing 539: noAdditionalIndent for displayMath in Version 3.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>noAdditionalIndent:</td>
</tr>
<tr>
<td>displayMath: 1</td>
</tr>
</tbody>
</table>

---

End
Index

— B —
backup files
  cycle through, 26
  extension settings, 25
  maximum number of backup files, 26
  number of backup files, 26
  overwrite switch, -w, 14

— C —
capturing parenthesis (regex), 39

— D —
delimiters, 109
  advanced settings, 31
  advanced settings of lookForAlignDelims, 30
  ampersand &, 31
  default settings of lookForAlignDelims, 31
  delimiter justification (left or right), 39
  delimiterRegEx, 39
  dontMeasure feature, 37
  double backslash demonstration, 36
  lookForAlignDelims, 31
  poly-switches for double back slash, 108
  spacing demonstration, 32
  with specialBeginEnd and the -m switch, 109
  within specialBeginEnd blocks, 45

— E —
exit code, 20
  summary, 20

— I —
indentation
  customising indentation per-code block, 48
  customising per-name, 48
  default, 17
  defaultIndent description, 30
  defaultIndent using -y switch, 17
  defaultIndent using YAML file, 21
  maximum indetation, 47
  no additional indent, 48
  no additional indent global, 48
  removing indentation per-code block, 48
  summary, 65

— L —
linebreaks
  summary of poly-switches, 109

— M —
modifying linebreaks

at the beginning of a code block, 101
at the end of a code block, 103
by text wrapping, globally, 73
by text wrapping, per-code-block, 76
by using one sentence per line, 91
surrounding double back slash, 108
using poly-switches, 100

— P —
poly-switches
  adding blank lines (again!): set to 4, 102, 104
  adding blank lines: set to 3, 102, 103
  adding comments and then line breaks: set to
    2, 101, 103
  adding line breaks: set to 1, 101, 103
  blank lines, 106
  conflicting partnering, 113
  conflicting switches, 114, 115
  default values, 100
  definition, 100
  double backslash, 109
  environment global example, 100
  environment per-code block example, 100
  for double back slash (delimiters), 108–111
  off by default: set to 0, 100
  removing line breaks: set to -1, 105
  summary of all poly-switches, 111
  values, 100
  visualisation: ♠, ♥, ♦, ♣, 101

— R —
regular expressions
  a word about, 12
  ampersand alignment, 31
  arguments, 132
  at least one +, 45, 119, 122, 132–134
  capturing parenthesis, 39
  character class demonstration, 136
  commands, 132
  delimiter alignment for edge or node, 45
  delimiter regex at #, 40
  delimiter regex at # or \>, 41
  delimiter regex at \= or \>, 39
  delimiter regex at only \>, 40
  delimiterRegEx, 31
  dontMeasure feature, cell, 38
  dontMeasure feature, row, 39
  environments, 132
  fine tuning, 132
  horizontal space \h, 45, 48, 99, 122, 123, 132
-m demonstration, 72–78, 80, 82–84, 86–88, 92–111, 113–116, 121
-n, –lines definition and details, 20
-o demonstration, 36, 40, 41, 45, 73–76, 82–84, 86–88, 123, 148
-r demonstration, 117–124
-rr demonstration, 120, 123
-rr, –onlyreplacement definition and details, 19
-rv demonstration, 123
-rrv, –replacementrespectverb definition and details, 19
-s, –silent definition and details, 16
-sl, –screenlog definition and details, 18
-t, –trace definition and details, 16
-tt, –ttrace definition and details, 16
-v, –version definition and details, 14
-w, –overwrite definition and details, 14
-y demonstration, 23, 36, 99
-y, –yaml definition and details, 17
— T —
text wrap
quick start, 73
recommended starting point, 89
— V —
verbatim
commands, 27
comparison with -r and -rr switches, 123
environments, 27
in relation to oneSentencePerLine, 97
in relation to paragraphsStopAt, 86
in relation to textWrapOptions, 74
noIndentBlock, 27
poly-switch summary, 111
rv, replacementrespectverb switch, 19, 117
specialBeginEnd, 44
verbatimEnvironments demonstration (-l switch), 23
verbatimEnvironments demonstration (-y switch), 23
within summary of text wrapping, 91
— W —
warning
amalgamate field, 70
be sure to test before use, 10
capturing parenthesis for lookForAlignDelims, 39
changing huge (textwrap), 75
ing environments, 27
editing YAML files, 22
fine tuning, 132
the m switch, 72

— S —
sentences
begin with, 93, 94end with, 93, 95
follow, 93
indenting, 98
one sentence per line, 91
oneSentencePerLine, 91
removing sentence line breaks, 92
text wrapping, 98
specialBeginEnd
alignment at delimiter, 45
combined with lookForAlignDelims, 45
default settings, 42
delimiterRegEx, 45
delimiterRegEx tweaked, 45
double backslash poly-switch demonstration, 109
IfElsFi example, 43
indentRules example, 61
indentRulesGlobal, 65
introduction, 42
lookForAlignDelims, 109
middle, 43
noAdditionalIndent, 61
noAdditionalIndentGlobal, 65
paragraphsStopAt, 86
poly-switch summary, 111
removeParagraphLineBreaks, 82
searching for special before commands, 42
specifying as verbatim, 44
textWrapOptions, 76
tikz example, 45
update to displaymath V3.0, 148

— T —
text wrap
quick start, 73
recommended starting point, 89
— V —
verbatim
commands, 27
comparison with -r and -rr switches, 123
environments, 27
in relation to oneSentencePerLine, 97
in relation to paragraphsStopAt, 86
in relation to textWrapOptions, 74
noIndentBlock, 27
poly-switch summary, 111
rv, replacementrespectverb switch, 19, 117
specialBeginEnd, 44
verbatimEnvironments demonstration (-l switch), 23
verbatimEnvironments demonstration (-y switch), 23
within summary of text wrapping, 91
— W —
warning
amalgamate field, 70
be sure to test before use, 10
capturing parenthesis for lookForAlignDelims, 39
changing huge (textwrap), 75
ing environments, 27
editing YAML files, 22
fine tuning, 132
the m switch, 72

— S —
sentences
begin with, 93, 94end with, 93, 95
follow, 93
indenting, 98
one sentence per line, 91
oneSentencePerLine, 91
removing sentence line breaks, 92
text wrapping, 98
specialBeginEnd
alignment at delimiter, 45
combined with lookForAlignDelims, 45
default settings, 42
delimiterRegEx, 45
delimiterRegEx tweaked, 45
double backslash poly-switch demonstration, 109
IfElsFi example, 43
indentRules example, 61
indentRulesGlobal, 65
introduction, 42
lookForAlignDelims, 109
middle, 43
noAdditionalIndent, 61
noAdditionalIndentGlobal, 65
paragraphsStopAt, 86
poly-switch summary, 111
removeParagraphLineBreaks, 82
searching for special before commands, 42
specifying as verbatim, 44
textWrapOptions, 76
tikz example, 45
update to displaymath V3.0, 148

— T —
text wrap
quick start, 73
recommended starting point, 89
— V —
verbatim
commands, 27
comparison with -r and -rr switches, 123
environments, 27
in relation to oneSentencePerLine, 97
in relation to paragraphsStopAt, 86
in relation to textWrapOptions, 74
noIndentBlock, 27
poly-switch summary, 111
rv, replacementrespectverb switch, 19, 117
specialBeginEnd, 44
verbatimEnvironments demonstration (-l switch), 23
verbatimEnvironments demonstration (-y switch), 23
within summary of text wrapping, 91
— W —
warning
amalgamate field, 70
be sure to test before use, 10
capturing parenthesis for lookForAlignDelims, 39
changing huge (textwrap), 75
ing environments, 27
editing YAML files, 22
fine tuning, 132
the m switch, 72