CODE DIFF TO PDF

Compared Two Versions of Source Code Listings

Author - me@example.com

Sample

Diff Compiled on September 9, 2019

Contents

1 Introduction

This is an example file that include source code listings from a text file. Red means that the code was deleted. Blue means that code was added.

2 src2pdf.tcl

File path: /home/betsalel/CygwinDocuments/code-diff-to-pdf/src2pdf.tcl

```
#!/usr/bin/tclsh
# Filename: src2pdf.tcl
# Copyright (c) 2019, Betsalel (Saul) Williamson, Jordan Henderson (the Authors)
# All rights reserved.
#
# Redistribution and use in source and binary forms, with or without
# modification, are permitted provided that the following conditions are met:
#     * Redistributions of source code must retain the above copyright
#       notice, this list of conditions and the following disclaimer.
#     * Redistributions in binary form must reproduce the above copyright
#       notice, this list of conditions and the following disclaimer in the
#       documentation and/or other materials provided with the distribution.
#     * Neither the names of the Authors nor the
#       names of its contributors may be used to endorse or promote products
#       derived from this software without specific prior written permission.
#
# THIS SOFTWARE IS PROVIDED BY THE Authors ``AS IS'' AND ANY
# EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
# WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
# DISCLAIMED. IN NO EVENT SHALL THE Authors BE LIABLE FOR ANY
# DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
# (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
# LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
# ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
# (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
# SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
#
#
# Notes:
#  2019-08-08 Initial creation
#  2019-08-12 Ran dos2unix on latex template and code example files
#  2019-08-18 Added removal or keeping of tmp folder with flag `-t'
#           Cleaned up line breaks in caption text generation
#           Added option to output to file with flag `-o fileName'
#           Cleaned up comments and other misc visual things in script
#           Added output for progress tracking
#           Renamed option variables
#  2019-08-30 Added input from text file to allow versions of code to be compared.
#           Fixed issue with calling script from other locations.
#  2019-08-30.2 Changed logic for generating added, removed, and change paths.
#  2019-09-08 Changed file name to src2pdf.tcl from diff-src.tcl
#           Added clean up proc to handle removing output in case of early exit
#
#
# the purpose for this diff tool:
# take as input files of version X (previous) and of version Y (current)
# build latex files with sources as listings
# run latexpand to inline sources
# run latexdiff on the latex files version X to Y
```

```
50 # build and output the diff as a PDF
51 # exit code 0 success
52 # exit code 1 error, no PDF
53 # exit code 2 warnings, PDF generated, but there may be issues
54 # exit code 3 script terminated unexpectedly
55
56 # assumptions:
57 # 1. files will follow their extensions and the syntax highlighting will not care about RPL
58 # 2. the resultant pdf will have content in the format: title page, TOC, listings with diffs
59 # 3. the input file contains pairs of files with the first file being the previous version
60 # 4. blank lines in the input file means the file was added or removed
61 #
62 #
63 # to find files: find . -regex ".*[.][h]?[st]?[cjmq]?[csl]"
64 #
65 # features that would be nice to have:
66 # parse headings of file (i.e. license info) to remove common headings and display in introduction
67 # have a separate TOC of diffs
68 # list of acronymns / dictionary used in code
69 # rolling change list sorted by date
70 # support for comparisons for more than one version
71 # choosing changes by major, minor, or build numbers
72 # parsing out comments into preabmles
73 # combination with Arros system to append figures and descriptions connected with source code
74 # add user information about the changes
75 # compare with diffs and blame for Fossil, Git and other popular systems
76 # right now, this tool is a stepping stone towards generating a PDF version of the complete system
       modeled in Arros
77 # we want to see from version to version what source code is being changed without care for who did
       it
78 # in the future, when users are integrated into the system, it will be possible to concern
       ourselves with tracking who inputs what
79 # preamble in latex file
80 # don't include titles and latex things that aren't needed (bib, list of tables and figures)
81 # run Expect spawn on PDF generation to interact with Latex and output latexdiff command
82 # explore the difference between running pdflatex and latexmk, use regular latexdiff and then run
       latexmk
83 #
84 # Common errors with latexdiff and latex:
85 # LaTeX Error: File `ulem.sty' not found.
86 #  missing tex package -> install with `tlmgr install PGK'
87 #                -> For SUSE linux use YaST. LaTex packages are prefixed with `texlive'
88 # WARNING: Inconsistency in length of input string and parsed string
89 #  Issue on Suse linux platform with texlive installed
90 #  latexdiff at older version. Update to version >= 1.3.
91 #   Download and install from https://www.ctan.org/tex-archive/support/latexdiff
92 # Error with `src2pdf.tcl' as input due to puts $fp "\\end\{lstlisting\}"
93 #  Latex assumed end of listing when the `{'s were not escaped
94 #
95 # see https://blog.tcl.tk/1246 for Tcllib installation
96 package require Tcl 8.6
97 package require cmdline
98 package require fileutil
99
100 set exitCode 3;# default unexpected exit
101
102 set usage ": src2pdf.tcl \[-d\] -i source-list-file.txt \[-o path-to-output.pdf\] \[-t\] \[-v\]\
       noptions:"
103
```

```
104 set parameters {
105    {i.arg  ""            "Required argument. Source list file. Contains list of code files. Relative
           paths must be relative to the file.\n\tFor regular input, each line should contain relative
           paths to source files.\n\tFor diff file, the format of file is that odd lines contain the
          previous versions of the source code file and even lines contain the next version. Empty
          lines mean that the file was removed or added.\n\tDefault:"}
106    {o.arg  "./output.pdf"  "Optional. Path and filename for result.\n\tDefault: "}
107    {d                 "Use diff for input source list file."}
108    {v              "Optional. Verbose output."}
109    {t              "Optional. Keep temporary Latex output."}
110 }
111
112 if {[catch {array set options [cmdline::getoptions ::argv $parameters $usage]}]} {
113     puts [cmdline::usage $parameters $usage]
114     exit $exitCode
115 } else {
116
117    if { $options(v) } {
118       puts "Verbose mode on."
119       set verboseOutput 1
120        parray options
121    } else {
122       set verboseOutput 0
123    }
124
125    if { $options(t) } {
126       puts "Keeping temporary LaTeX output."
127       set keepTmpOutput 1
128    } else {
129       set keepTmpOutput 0
130    }
131
132    if { $options(d) } {
133       puts "Performing code src diff to pdf."
134       set diffMode 1
135    } else {
136       puts "Performing regular src to pdf."
137       set diffMode 0
138    }
139
140    if {[string length $options(o)] > 0} {
141       set outputFile [file normalize $options(o)]
142       puts "Output file set:\n$outputFile"
143    } else {
144       set outputFile [file normalize "./output.pdf"]
145    }
146
147    if {[string length $options(i)] > 0} {
148       set inputFile [file normalize $options(i)]
149       puts "Input file set:\n$inputFile"
150    } else {
151       puts "Missing source list file.\n\n"
152        puts [cmdline::usage $parameters $usage]
153       exit 1
154    }
155 }
156
157 proc isLink { aFile } {
158     return [expr ! [catch {file readlink ${aFile}}]]
```

```
159 }
160
161 if { ! ${tcl_interactive} } {
162     ;# not being interactive is not enough!
163     ;# we could be being included which makes us a script
164     ;# in which case argv0 would not apply
165     set ::myName [info script]
166     ;#set myName ${argv0}
167 } else {
168     set ::myName [info script]
169 }
170 if { [isLink ${::myName}] } {
171     set ::myName [file readlink ${::myName}]
172 }
173 set ::myName [file normalize ${::myName}]
174 while {! [file isdirectory $::myName] && ! [catch {file readlink ${::myName}} newName] } {
175     set ::myName ${newName}
176 }
177 if { ! [file isdirectory ${::myName}] } {
178     set ::myBase [file dirname ${::myName}]
179 } else {
180     set ::myBase ${::myName}
181 }
182
183 if { $verboseOutput } {
184    puts "Base:\n${::myBase}"
185 }
186
187 proc printList {list} {
188    puts "Total files: [llength $list]"
189
190    foreach elem $list {
191       if {[string length $elem] > 0} {
192          puts "Opening $elem ..."
193          set fp [open $elem r]
194          set fileData [read $fp]
195          puts $fileData
196          close $fp
197          puts "Closed $elem\n\n"
198       }
199    }
200
201    return
202 }
203
204 puts -nonewline "Preparing LaTeX files...     "
205 flush stdout
206
207 proc prepTempDir {} {
208    global ::myBase
209    global diffMode
210
211    ;# del prev, set magic string to avoid collision with existing dir for rm -rf
212    ;# TODO redo magic number if there was a collision and the tmp dir with
213    ;# that number already exists.
214    set magicNumber  "latex-template-[expr {int(rand()*999999999999) + 1}]"
215    set tmpDir      "tmp-$magicNumber"
216    exec rm -rf $tmpDir ;# this is dangerous if $tmpDir is set to another directory
217    exec mkdir -p $tmpDir
```

```
218
219    set latexPrevDir  "$tmpDir/latex-prev/"
220    set latexCurDir  "$tmpDir/latex-cur/"
221
222    if { $diffMode } {
223       ;# create latex and reference locations
224       exec cp -R ${::myBase}/latex-template $latexPrevDir
225       exec cp -R ${::myBase}/latex-template $latexCurDir
226    } else {
227       exec cp -R ${::myBase}/latex-template $tmpDir
228    }
229
230    return [list [file normalize $latexPrevDir] [file normalize $latexCurDir] [file normalize
           $tmpDir]];
231 }
232
233 set latexRoots  [prepTempDir]
234 set prevRoot  [lindex $latexRoots 0]
235 set curRoot   [lindex $latexRoots 1]
236 set tmpDir    [lindex $latexRoots 2]
237
238 proc cleanUp {keepTmpOutput} {
239    global tmpDir
240
241    if {[string length $tmpDir] > 0} {
242       if { $keepTmpOutput } {
243          puts "Keeping temporary LaTeX output at:\n$tmpDir"
244       } else {
245          puts -nonewline "Removing temporary LaTeX output..."
246          flush stdout
247          exec rm -r $tmpDir
248          puts " Done."
249       }
250    }
251 }
252
253 proc escapePathForLatex {s} {
254    global verboseOutput
255
256    if { $verboseOutput } {
257       puts "String before escaping: "
258       puts $s
259    }
260
261    ;# special characters in latex must be escaped
262     ;# \ { } & ^ $ % ;# _ ~
263
264    regsub -all {([\\{\}&\^\$%#_~])} $s {\\1} s
265
266    ;# add latex line breaks into paths
267    regsub -all {([\\]?[\/&\^\$%#_~\-;'!])} $s {\1\\discretionary{}{}{}} s
268
269    if { $verboseOutput } {
270       puts "String after escaping: "
271       puts $s
272    }
273    return $s
274 }
275
```

```
276 proc getLanguageFromFile {fileName} {
277    global verboseOutput
278 
279    set fe [file extension $fileName]
280    if {  $verboseOutput } {
281       puts "File extension: $fe"
282    }
283 
284    set language ""
285    switch $fe {
286       .c {
287          set language "C"
288       }
289       .sql {
290          set language "SQL"
291       }
292       .js {
293          set language "ECMAScript"
294       }
295       .tcl {
296          set language "tcl"
297       }
298       default {}
299    }
300 
301    return $language
302 }
303 
304 proc addCodeToFile {texFilePath sectionText bodyText codeFilePath} {
305    global verboseOutput
306 
307    set fp [open $texFilePath a+]
308    puts $fp "\\section\{$sectionText\}"
309    puts $fp "$bodyText"
310    ;# puts $fp "\\label{sec:$captionText}"
311    ;# label=code:$fileName
312    ;# TODO ensure that the equivalent files have the same label to allow references to work
       correctly
313    set fp1 [open $codeFilePath r]
314    set fileData [read $fp1]
315    close $fp1
316    puts $fp "\\begin\{lstlisting\}\[language=[getLanguageFromFile $codeFilePath]\]"
317    puts $fp $fileData
318    puts $fp "\\end\{lstlisting\}"
319    close $fp
320 
321    if {  $verboseOutput } {
322       puts "Wrote data into tex file..."
323    }
324    return;
325 }
326 
327 proc processDiffInputFile {inputFile prevRoot curRoot} {
328    global verboseOutput
329 
330    if {  $verboseOutput } {
331       puts "\nProcessing input file..."
332    }
333    if {[catch {exec dos2unix -q $inputFile} result]} {
```

```
334       puts "Error with converting line endings to Unix."
335          puts "Information about error: $::errorInfo"
336          puts $result
337          cleanUp 1
338          exit 1
339    }
340
341    set fpInputFile [open $inputFile]
342    set inputFileLines [split [read $fpInputFile] "\n"]
343    if {  $verboseOutput } {
344       puts "File lines: [llength $inputFileLines]"
345       puts "Input files $inputFileLines"
346    }
347
348    close $fpInputFile;
349
350    ;# Change path to the input file source to ensure relative paths work
351    set curDir [exec pwd];
352    cd [file dirname $inputFile]
353
354    for { set i 0} {$i < [llength $inputFileLines]} {incr i 2} {
355       ;# check to see if file was added/removed
356
357       set norm1 [lindex $inputFileLines $i]
358       set norm1 [file normalize $norm1]
359
360       set norm2 [lindex $inputFileLines [expr $i+1]]
361       set norm2 [file normalize $norm2]
362
363       ;## both norm1 and norm2 cant be empty
364       set norm1Empty [expr ![string length $norm1]]
365       set norm2Empty [expr ![string length $norm2]]
366
367       if {$norm1Empty && $norm2Empty} {
368          puts "Error with input file. Cannot have two blank lines $i and [expr $i+1]."
369             cleanUp 1
370          exit 1
371       }
372
373       if {!($norm1Empty)} {
374          set prevPathForLatex [escapePathForLatex $norm1]
375          set prevFileName [file tail $norm1]
376       }
377
378       if {!($norm2Empty)} {
379          set curPathForLatex [escapePathForLatex $norm2]
380          set curFileName [file tail $norm2]
381       }
382
383       ;# if added grab second line and insert to first that this was added
384       if {$norm1Empty} {
385          if {  $verboseOutput } {
386             puts "added:\n$norm2"
387          }
388          ;# prev
389          addCodeToFile "$prevRoot/body/01-code.tex" "Added $curFileName" "New file: $curPathForLatex
                 " "$norm2"
390          ;# cur
391          addCodeToFile "$curRoot/body/01-code.tex" "Added $curFileName" "New file: $curPathForLatex"
```

```
            "$norm2"

      ;# if removed grab first line and insert to next that this was removed
      } elseif {$norm2Empty} {
         if { $verboseOutput } {
            puts "removed:\n$norm1"
         }
         ;# prev
         addCodeToFile "$prevRoot/body/01-code.tex" "Removed $prevFileName" "Removed file:
             $prevPathForLatex" "$norm1"
         ;# cur
         addCodeToFile "$curRoot/body/01-code.tex" "Removed $prevFileName" "Removed file:
             $prevPathForLatex" "$norm1"

      ;# if directory path doesn't equal then add change path
      } elseif {[string compare [file dirname $norm1] [file dirname $norm2]] != 0} {
         if { $verboseOutput } {
            puts "path changed:\n$norm1\n$norm2"
         }
         ;# prev
         addCodeToFile "$prevRoot/body/01-code.tex" "Changed Path $prevFileName" "Changed file path
             from: $prevPathForLatex to $curPathForLatex" "$norm1"
         ;# cur
         addCodeToFile "$curRoot/body/01-code.tex" "Changed Path $curFileName" "Changed file path
             from: $prevPathForLatex to $curPathForLatex" "$norm2"
      } else {
         ;# prev
         addCodeToFile "$prevRoot/body/01-code.tex" "$prevFileName" "File path: $prevPathForLatex" "
             $norm1"
         ;# cur
         addCodeToFile "$curRoot/body/01-code.tex" "$curFileName" "File path: $curPathForLatex" "
             $norm2"
      }
   }

   cd $curDir

   if { $verboseOutput } {
      puts "Finished looping through files..."
   }

   return;
}

proc processRegularInputFile {inputFile tmpDir} {
   global verboseOutput

   if { $verboseOutput } {
      puts "\nProcessing input file..."
   }
   if {[catch {exec dos2unix -q $inputFile} result]} {
      puts "Error with converting line endings to Unix."
         puts "Information about error: $::errorInfo"
         puts $result
         cleanUp 1
         exit 1
   }

   set fpInputFile [open $inputFile]
```

```
444     set inputFileLines [split [read $fpInputFile] "\n"]
445     if { $verboseOutput } {
446        puts "File lines: [llength $inputFileLines]"
447        puts "Input files $inputFileLines"
448     }
449
450     close $fpInputFile;
451
452     ;# Change path to the input file source to ensure relative paths work
453     set curDir [exec pwd];
454     cd [file dirname $inputFile]
455
456     for { set i 0} {$i < [llength $inputFileLines]} {incr i} {
457        ;# check to see if file was added/removed
458
459        set norm1 [lindex $inputFileLines $i]
460        set norm1 [file normalize $norm1]
461
462        ;## both norm1 and norm2 cant be empty
463        set norm1Empty [expr ![string length $norm1]]
464
465        if {$norm1Empty} {
466           continue;
467        }
468
469        set pathForLatex [escapePathForLatex $norm1]
470        set fileName [file tail $norm1]
471        addCodeToFile "$tmpDir/latex-template/body/01-code.tex" "$fileName" "File path: $pathForLatex"
               "$norm1"
472     }
473
474     cd $curDir
475
476     if { $verboseOutput } {
477        puts "Finished looping through files..."
478     }
479
480     return;
481  }
482
483  if { $diffMode } {
484     processDiffInputFile $inputFile $prevRoot $curRoot
485  } else {
486     processRegularInputFile $inputFile $tmpDir
487  }
488
489  proc inlineLatex {latexRoot} {
490     global verboseOutput
491     set prevDir [pwd]
492     set inlineMain  "main2.tex"
493     set defaultMain "main.tex"
494     set defaultBib  "references.bib"; ;#TODO still not sure if this needs to be a bbl or bib file
495     cd $latexRoot
496     exec latexpand --keep-comments --expand-bbl $defaultBib $defaultMain > $inlineMain
497     if { $verboseOutput } {
498        puts "latexpand --keep-comments --expand-bbl $defaultBib $defaultMain > $inlineMain"
499     }
500
501     cd $prevDir
```

```
502     return $inlineMain;
503 }
504 
505 if { $diffMode } {
506     ;# inline latex to make it easier to do the diff
507     set inline  [inlineLatex $prevRoot]
508     set inline2 [inlineLatex $curRoot]
509 } else {
510     set inline  [inlineLatex "$tmpDir/latex-template"]
511 }
512 
513 if {  $verboseOutput } {
514     puts "Finished inlining files..."
515 }
516 
517 puts " Done."; ;# preparing latex files
518 
519 # generate pdf
520 # uses latexdiff-vc run in the temp folder
521 
522 cd "$tmpDir"
523 
524 # this was the original diff latex, but produces latex output
525 # the following program latexdiff-vc is preferred because it creates direct to PDF
526 # manual diff to .tex
527 # exec latexdiff "$prevRoot/$inline" "$curRoot/$inline2" > "$tmpDir/main2-diff.tex"
528 # spawn latexdiff-vc --verbose --pdf "$prevRoot/$inline" "$curRoot/$inline2"
529 
530 puts "If the following step hangs (more than 2 minutes) enter 'x' and hit enter."
531 puts -nonewline "Generating PDF...             "
532 flush stdout
533 
534 if { $diffMode } {
535     ;# inline latex to make it easier to do the diff
536     ;# TODO investigate if there will be security issues with input sources being malicious from TCL
          or Latex
537     set cmd "latexdiff-vc --pdf \"$prevRoot/$inline\" \"$curRoot/$inline2\""
538     set tmpOutputPdf "main2-diff.pdf"
539 } else {
540     set cmd "latexmk -pdf \"$tmpDir/latex-template/$inline\""
541     set tmpOutputPdf "main2.pdf"
542 }
543 
544 
545 if { $verboseOutput } {
546     puts "Running command to generate PDF:"
547     puts $cmd
548 }
549 
550 if {[catch {exec {*}$cmd} result]} {
551 
552     ;# report errors and warnings,
553     ;# ideally there shouldn't be any because we are generating everything here
554     ;# but LaTex generation has many warnings that are a nuisance
555     ;# input code files shouldn't break anything
556     ;# todo change to expect script to allow interaction with above program in case something
          happens
557     ;# as of now the program halts if there is a latex warning that requries user engagement
558       puts "\nInformation about error: $::errorInfo\n\n"
```

```
559
560     if {[file exists "$tmpDir/$tmpOutputPdf"]} {
561       puts "Warning on pdf generation!"
562       set exitCode 2
563     } else {
564       puts stderr "Error!"
565       set exitCode 1
566     }
567
568 } else {
569     set exitCode 0
570 }
571
572 puts " Done."; ;# generating PDF
573
574 if {[file exists "$tmpDir/$tmpOutputPdf"]} {
575
576    exec mv "$tmpDir/$tmpOutputPdf" $outputFile
577
578    if {[file exists "$outputFile"]} {
579       puts "PDF created at:\n$outputFile"
580       set exitCode 0
581    } else {
582       puts "Error generating output..."
583       set exitCode 1
584    }
585
586 } else {
587    puts "Error generating output..."
588    set exitCode 1
589 }
590
591 cleanUp $keepTmpOutput
592
593 exit $exitCode
```

3 tcl-file.tcl

File path: /home/betsalel/CygwinDocuments/code-diff-to-pdf/example/v2/folder with space at end/tcl-file.tcl

```
1 #!/usr/bin/tclsh
2
3 puts "Hello World"
4 # example comment
5
6 puts "A very long line of text. A very long line of text. A very long line of text. A very long
      line of text. A very long line of text. A very long line of text. A very long line of text. A
      very long line of text. A very long line of text. A very long line of text. A very long line of
       text. A very long line of text."
7
8 # the escaped quotes should not end the quote
9 set cmd "latexdiff-vc --pdf \"$prevRoot/$inline\" \"$curRoot/$inline2\""
```

4 js-file.js

File path: /home/betsalel/CygwinDocuments/code-diff-to-pdf/example/v2/folder with space at end/js-file.js

```
console.log( 'Hello, world!' );
// JUST FOR THE HECK OF IT! I'LL CHANGE TOO :-)
var foo = "some \" string with a quote\' ";

// the regex delimeters should be highlighted differently
var regex = foo.toLowerCase().replace(/[ ,.\/!@#$%\^&*\(\)\-_,+=|\\\{\}\[\]\`~:;\"\'<>?]{1,}/g,'-')
    ;
```

5 sql-file.sql

File path: /home/betsalel/CygwinDocuments/code-diff-to-pdf/example/v2/folder$with# Chars%;'!&/sql-file.sql

```
DROP TABLE HELLO_WORLD;

CREATE TABLE HELLO_WORLD(
   ID   INT              NOT NULL,
   PRIMARY KEY (ID)
);
```