

CODE DIFF TO PDF

---

# Compared Two Versions of Source Code Listings

---

Author - me@example.com

Sample

Diff Compiled on September 8, 2019

## Contents

1	Introduction	2
2	src2pdf.tcl	2
3	tcl-file.tcl	12
4	js-file.js	13
5	sql-file.sql	13

# 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

```

1  #!/usr/bin/tclsh
2  # Filename: src2pdf.tcl
3  # Copyright (c) 2019, Betsalel (Saul) Williamson, Jordan Henderson (the Authors)
4  # All rights reserved.
5  #
6  # Redistribution and use in source and binary forms, with or without
7  # modification, are permitted provided that the following conditions are met:
8  #     * Redistributions of source code must retain the above copyright
9  #       notice, this list of conditions and the following disclaimer.
10 #     * Redistributions in binary form must reproduce the above copyright
11 #       notice, this list of conditions and the following disclaimer in the
12 #       documentation and/or other materials provided with the distribution.
13 #     * Neither the names of the Authors nor the
14 #       names of its contributors may be used to endorse or promote products
15 #       derived from this software without specific prior written permission.
16 #
17 # THIS SOFTWARE IS PROVIDED BY THE Authors ‘‘AS IS’’ AND ANY
18 # EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
19 # WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
20 # DISCLAIMED. IN NO EVENT SHALL THE Authors BE LIABLE FOR ANY
21 # DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
22 # (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
23 # LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
24 # ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
25 # (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
26 # SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
27 #
28 #
29 # Notes:
30 # 2019-08-08 Initial creation
31 # 2019-08-12 Ran dos2unix on latex template and code example files
32 # 2019-08-18 Added removal or keeping of tmp folder with flag ‘-t’
33 #     Cleaned up line breaks in caption text generation
34 #     Added option to output to file with flag ‘-o fileName’
35 #     Cleaned up comments and other misc visual things in script
36 #     Added output for progress tracking
37 #     Renamed option variables
38 # 2019-08-30 Added input from text file to allow versions of code to be compared.
39 #     Fixed issue with calling script from other locations.
40 # 2019-08-30.2 Changed logic for generating added, removed, and change paths.
41 # 2019-09-08 Changed file name to src2pdf.tcl from diff-src.tcl
42 #     Added clean up proc to handle removing output in case of early exit
43 #
44 #
45 # the purpose for this diff tool:
46 # take as input files of version X (previous) and of version Y (current)
47 # build latex files with sources as listings
48 # run latexexpand to inline sources
49 # 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]?[cjm]?[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 preambles
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
77   ↪ modeled in Arros
78 # we want to see from version to version what source code is being changed without care for who did
79   ↪ it
80 # in the future, when users are integrated into the system, it will be possible to concern
81   ↪ ourselves with tracking who inputs what
82 # preamble in latex file
83 # don't include titles and latex things that aren't needed (bib, list of tables and figures)
84 # run Expect spawn on PDF generation to interact with Latex and output latexdiff command
85 # explore the difference between running pdflatex and latexmk, use regular latexdiff and then run
86   ↪ latexmk
87 #
88 # Common errors with latexdiff and latex:
89 # LaTeX Error: File 'ulem.sty' not found.
90 # missing tex package -> install with 'tlmgr install PGK'
91 # -> For SUSE linux use YaST. LaTeX packages are prefixed with 'texlive'
92 # WARNING: Inconsistency in length of input string and parsed string
93 # Issue on Suse linux platform with texlive installed
94 # latexdiff at older version. Update to version >= 1.3.
95 # Download and install from https://www.ctan.org/tex-archive/support/latexdiff
96 # Error with 'src2pdf.tcl' as input due to puts $fp "\\end{\\lstlisting\\}"
97 # Latex assumed end of listing when the '{'s were not escaped
98 #
99 # see https://blog.tcl.tk/1246 for Tcllib installation
100 package require Tcl 8.6
101 package require cmdline
102 package require fileutil
103
104 set exitCode 3;# default unexpected exit
105
106 set usage ": src2pdf.tcl [-d] -i source-list-file.txt [-o path-to-output.pdf] [-t] [-v]\\
107   ↪ noptions:"

```

```

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
313     ↪ correctly
314     set fp1 [open $codeFilePath r]
315     set fileData [read $fp1]
316     close $fp1
317     puts $fp "\\begin\\{lstlisting\\}\\[language=[getLanguageFromFile $codeFilePath]\\]"
318     puts $fp $fileData
319     puts $fp "\\end\\{lstlisting\\}"
320     close $fp
321
322     if { $verboseOutput } {
323         puts "Wrote data into tex file..."
324     }
325     return;
326 }
327
328 proc processDiffInputFile {inputFile prevRoot curRoot} {
329     global verboseOutput
330
331     if { $verboseOutput } {
332         puts "\nProcessing input file..."
333     }
334     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: $curPathForLatex
390             ↪ " "$norm2"
391         ;# cur
392         addCodeToFile "$curRoot/body/01-code.tex" "Added $curFileName" "New file: $curPathForLatex"

```

```

    ↪ "$norm2"
392
393     ;# if removed grab first line and insert to next that this was removed
394 } elseif {$norm2Empty} {
395     if { $verboseOutput } {
396         puts "removed:\n$norm1"
397     }
398     ;# prev
399     addCodeToFile "$prevRoot/body/01-code.tex" "Removed $prevFileName" "Removed file:
    ↪ $prevPathForLatex" "$norm1"
400
401     ;# cur
402     addCodeToFile "$curRoot/body/01-code.tex" "Removed $prevFileName" "Removed file:
    ↪ $prevPathForLatex" "$norm1"
403
404     ;# if directory path doesn't equal then add change path
405 } elseif {[string compare [file dir $norm1] [file dir $norm2]] != 0} {
406     if { $verboseOutput } {
407         puts "path changed:\n$norm1\n$norm2"
408     }
409     ;# prev
410     addCodeToFile "$prevRoot/body/01-code.tex" "Changed Path $prevFileName" "Changed file path
    ↪ from: $prevPathForLatex to $curPathForLatex" "$norm1"
411
412     ;# cur
413     addCodeToFile "$curRoot/body/01-code.tex" "Changed Path $curFileName" "Changed file path
    ↪ from: $prevPathForLatex to $curPathForLatex" "$norm2"
414 } else {
415     ;# prev
416     addCodeToFile "$prevRoot/body/01-code.tex" "$prevFileName" "File path: $prevPathForLatex" "
    ↪ $norm1"
417
418     ;# cur
419     addCodeToFile "$curRoot/body/01-code.tex" "$curFileName" "File path: $curPathForLatex" "
    ↪ $norm2"
420 }
421 }
422
423 cd $curDir
424
425 if { $verboseOutput } {
426     puts "Finished looping through files..."
427 }
428
429 return;
430 }
431
432 proc processRegularInputFile {inputFile tmpDir} {
433     global verboseOutput
434
435     if { $verboseOutput } {
436         puts "\nProcessing input file..."
437     }
438     if {[catch {exec dos2unix -q $inputFile} result]} {
439         puts "Error with converting line endings to Unix."
440         puts "Information about error: $::errorInfo"
441         puts $result
442         cleanup 1
443         exit 1
444     }
445
446     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         puts "Error with input file. Cannot have blank line $i."
467         exit 1
468     }
469
470     set pathForLatex [escapePathForLatex $norm1]
471     set fileName [file tail $norm1]
472     addCodeToFile "$tmpDir/latex-template/body/01-code.tex" "$fileName" "File path: $pathForLatex"
473         ↪ "$norm1"
474 }
475
476 cd $curDir
477
478 if { $verboseOutput } {
479     puts "Finished looping through files..."
480 }
481
482 return;
483 }
484
485 if { $diffMode } {
486     processDiffInputFile $inputFile $prevRoot $curRoot
487 } else {
488     processRegularInputFile $inputFile $tmpDir
489 }
490
491 proc inlineLatex {latexRoot} {
492     global verboseOutput
493     set prevDir [pwd]
494     set inlineMain "main2.tex"
495     set defaultMain "main.tex"
496     set defaultBib "references.bib"; ;#TODO still not sure if this needs to be a bbl or bib file
497     cd $latexRoot
498     exec latexexpand --keep-comments --expand-bbl $defaultBib $defaultMain > $inlineMain
499     if { $verboseOutput } {
500         puts "latexexpand --keep-comments --expand-bbl $defaultBib $defaultMain > $inlineMain"
501     }

```

```

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

```

```
559     puts "\nInformation about error: $::errorInfo\n\n"
560
561     if {[file exists "$tmpDir/$tmpOutputPdf"]} {
562         puts "Warning on pdf generation!"
563         set exitCode 2
564     } else {
565         puts stderr "Error!"
566         set exitCode 1
567     }
568
569 } else {
570     set exitCode 0
571 }
572
573 puts " Done."; ;# generating PDF
574
575 if {[file exists "$tmpDir/$tmpOutputPdf"]} {
576
577     exec mv "$tmpDir/$tmpOutputPdf" $outputFile
578
579     if {[file exists "$outputFile"]} {
580         puts "PDF created at:\n$outputFile"
581         set exitCode 0
582     } else {
583         puts "Error generating output..."
584         set exitCode 1
585     }
586
587 } else {
588     puts "Error generating output..."
589     set exitCode 1
590 }
591
592 cleanUp $keepTmpOutput
593
594 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."
```

---

## 4 js-file.js

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

---

```
1 console.log( 'Hello, world!' );
2 // JUST FOR THE HECK OF IT! I'LL CHANGE TOO :-)
```

---

## 5 sql-file.sql

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

---

```
1 DROP TABLE HELLO_WORLD;
2
3 CREATE TABLE HELLO_WORLD(
4     ID INT NOT NULL,
5     PRIMARY KEY (ID)
6 );
```

---