

CODE DIFF TO PDF

Compared Two Versions of Source Code Listings

Author - me@example.com

Sample

Diff Compiled on August 31, 2019

Contents

1	Introduction	2
2	diff-src.tcl	2
3	Removed c file.c	10
4	Changed Path tcl-file.tcl	11
5	Changed Path js-file.js	11
6	Added sql-file.sql	11

Listings

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 diff-src.tcl

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

```
1 #!/usr/bin/tclsh
2 # Filename: diff-src.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 #
42 # the purpose for this diff tool:
43 # take as input files of version X (previous) and of version Y (current)
44 # build latex files with sources as listings
45 # run latexexpand to inline sources
46 # run latexdiff on the latex files version X to Y
47 # build and output the diff as a PDF
48 # exit code 0 success
49 # exit code 1 error, no PDF
```

```

50 # exit code 2 warnings, PDF generated, but there may be issues
51 # exit code 3 script terminated unexpectedly
52
53 # assumptions:
54 # 1. files will follow their extensions and the syntax highlighting will not care about RPL
55 # 2. the resultant pdf will have content in the format: title page, TOC, listings with diffs
56 # 3. the input file contains pairs of files with the first file being the previous version
57 # 4. blank lines in the input file means the file was added or removed
58 #
59 #
60 # to find files: find . -regex ".*[.][h]?[st]?[c]mq?[csl]"
61 #
62 # features that would be nice to have:
63 # parse headings of file (i.e. license info) to remove common headings and display in introduction
64 # have a separate TOC of diffs
65 # list of acronymms / dictionary used in code
66 # rolling change list sorted by date
67 # support for comparisons for more than one version
68 # choosing changes by major, minor, or build numbers
69 # parsing out comments into preambles
70 # combination with Arros system to append figures and descriptions connected with source code
71 # add user information about the changes
72 # compare with diffs and blame for Fossil, Git and other popular systems
73 # right now, this tool is a stepping stone towards generating a PDF version of the complete system
74   ↪ modeled in Arros
75 # we want to see from version to version what source code is being changed without care for who did
76   ↪ it
77 # in the future, when users are integrated into the system, it will be possible to concern
78   ↪ 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 #
83 # Common errors with latexdiff and latex:
84 # LaTeX Error: File 'ulem.sty' not found.
85 # missing tex package -> install with 'tlmgr install PGK'
86 # WARNING: Inconsistency in length of input string and parsed string
87 # Issue on Suse linux platform with texlive installed
88 # latexdiff at older version. Update to version >= 1.3.
89 # Download and install from https://www.ctan.org/tex-archive/support/latexdiff
90 # Error with 'diff-src.tcl' as input due to puts $fp "\\end{\\lstlisting\\}"
91 # Latex assumed end of listing when the '{'s were not escaped
92 #
93 # see https://blog.tcl.tk/1246 for Tcllib installation
94 package require Tcl 8.6
95 package require cmdline
96 package require fileutil
97
98 set exitCode 3;# default unexpected exit
99
100 set usage ": diff-src.tcl -i source-list-file.txt \\[-o path-to-output.pdf\\] \\[-t\\] \\[-v\\]\\noptions:
101   ↪ "
102
103 set parameters {
104   {i.arg "" "Required argument. Source list file. Contains list of code files. Relative
105     ↪ paths must be relative to the file. Format of file is that odd lines contain the
106     ↪ previous versions of the source code file and even lines contain the next version. Empty
107     ↪ lines mean that the file was removed or added.\\n\\tDefault:"}

```

```

102 {o.arg  "./output.pdf"  "Optional. Path and filename for result\n\tDefault: "}
103 {v      "Optional. Verbose output."}
104 {t      "Optional. Keep temporary Latex output."}
105 }
106
107 if {[catch {array set options [cmdline::getoptions ::argv $parameters $usage]}} {
108     puts [cmdline::usage $parameters $usage]
109     exit $exitCode
110 } else {
111
112     if { $options(v) } {
113         puts "Verbose mode on."
114         set verboseOutput 1
115         parray options
116     } else {
117         set verboseOutput 0
118     }
119
120     if { $options(t) } {
121         puts "Keeping temporary LaTeX output."
122         set keepTmpOutput 1
123     } else {
124         set keepTmpOutput 0
125     }
126
127     if {[string length $options(o)] > 0} {
128         set outputFile [file normalize $options(o)]
129         puts "Output file set:\n$outputFile"
130     } else {
131         set outputFile [file normalize "./output.pdf"]
132     }
133
134     if {[string length $options(i)] > 0} {
135         set inputFile [file normalize $options(i)]
136         puts "Input file set:\n$inputFile"
137     } else {
138         puts "Missing source list file.\n\n"
139         puts [cmdline::usage $parameters $usage]
140         exit 1
141     }
142 }
143
144 proc isLink { aFile } {
145     return [expr ! [catch {file readlink ${aFile}}]]
146 }
147
148 if { ! ${tcl_interactive} } {
149     # not being interactive is not enough!
150     # we could be being included which makes us a script
151     # in which case argv0 would not apply
152     set ::myName [info script]
153     #set myName ${argv0}
154 } else {
155     set ::myName [info script]
156 }
157 if { [isLink ${::myName}] } {
158     set ::myName [file readlink ${::myName}]
159 }
160 set ::myName [file normalize ${::myName}]

```

```

161 while {![file isdirectory $::myName] && ! [catch {file readlink $::myName} newName] } {
162     set ::myName ${newName}
163 }
164 if { ! [file isdirectory $::myName] } {
165     set ::myBase [file dirname $::myName]
166 } else {
167     set ::myBase $::myName
168 }
169
170 if { $verboseOutput } {
171     puts "Base:\n${::myBase}"
172 }
173
174 proc printList {list} {
175     puts "Total files: [llength $list]"
176
177     foreach elem $list {
178         if {[string length $elem] > 0} {
179             puts "Opening $elem ..."
180             set fp [open $elem r]
181             set fileData [read $fp]
182             puts $fileData
183             close $fp
184             puts "Closed $elem\n\n"
185         }
186     }
187
188     return
189 }
190
191 puts -nonewline "Preparing LaTeX files...      "
192 flush stdout
193
194 proc prepTempDir {} {
195     global ::myBase
196
197     # del prev, set magic string to avoid collision with existing dir for rm -rf
198     # TODO redo magic number if there was a collision and the tmp dir with
199     # that number already exists.
200     set magicNumber "latex-template-[expr {int(rand()*999999999999) + 1}]"
201     set tmpDir "tmp-$magicNumber"
202     exec rm -rf $tmpDir ;# this is dangerous if $tmpDir is set to another directory
203     exec mkdir -p $tmpDir
204     set latexPrevDir "$tmpDir/latex-prev/"
205     set latexCurDir "$tmpDir/latex-cur/"
206
207     # create latex and reference locations
208     exec cp -R $::myBase/latex-template $latexPrevDir
209     exec cp -R $::myBase/latex-template $latexCurDir
210
211     return [list [file normalize $latexPrevDir] [file normalize $latexCurDir] [file normalize
212         ↪ $tmpDir]];
213 }
214
215 set latexRoots [prepTempDir]
216 set prevRoot [lindex $latexRoots 0]
217 set curRoot [lindex $latexRoots 1]
218 set tmpDir [lindex $latexRoots 2]

```

```

219 proc escapePathForLatex {s} {
220     global verboseOutput
221
222     if { $verboseOutput } {
223         puts "String before escaping: "
224         puts $s
225     }
226
227     # special characters in latex must be escaped
228     # \ { } & ^ $ % # _ ~
229
230     regsub -all {[\\\{\}&^\$%#_~]} $s {[\\1]} s
231
232     # add latex line breaks into paths
233     regsub -all {[\\]?[\/&^\$%#_~\-\;'!]} $s {\1\\discretionary{}{}{}} s
234
235     if { $verboseOutput } {
236         puts "String after escaping: "
237         puts $s
238     }
239     return $s
240 }
241
242 proc getLanguageFromFile {fileName} {
243     global verboseOutput
244
245     set fe [file extension $fileName]
246     if { $verboseOutput } {
247         puts "File extension: $fe"
248     }
249
250     set language ""
251     switch $fe {
252         .c {
253             set language "C"
254         }
255         .sql {
256             set language "SQL"
257         }
258         .js {
259             set language "ECMAScript"
260         }
261         .tcl {
262             set language "tcl"
263         }
264         default {}
265     }
266
267     return $language
268 }
269
270 proc addCodeToFile {texFilePath sectionText bodyText codeFilePath} {
271     global verboseOutput
272
273     set fp [open $texFilePath a+]
274     puts $fp "\\section\\{$sectionText\\}"
275     puts $fp "$bodyText"
276     # puts $fp "\\label{sec:$captionText}"
277     # label=code:$fileName

```

```

278 # TODO ensure that the equivalent files have the same label to allow references to work
      ↪ correctly
279 set fp1 [open $codeFilePath r]
280 set fileData [read $fp1]
281 close $fp1
282 puts $fp "\\begin\\{lstlisting\\}\\[language=[getLanguageFromFile $codeFilePath]\\]"
283 puts $fp $fileData
284 puts $fp "\\end\\{lstlisting\\}"
285 close $fp
286
287 if { $verboseOutput } {
288     puts "Wrote data into tex file..."
289 }
290 return;
291 }
292
293 proc processInputFile {inputFile prevRoot curRoot} {
294     global verboseOutput
295
296     if { $verboseOutput } {
297         puts "\\nProcessing input file..."
298     }
299     if {[catch {exec dos2unix -q $inputFile} result]} {
300         puts "Error with converting line endings to Unix."
301         puts "Information about error: $::errorInfo"
302         puts $result
303         exit 1
304     }
305
306     set fpInputFile [open $inputFile]
307     set inputFileLines [split [read $fpInputFile] "\\n"]
308     if { $verboseOutput } {
309         puts "File lines: [llength $inputFileLines]"
310         puts "Input files $inputFileLines"
311     }
312
313     close $fpInputFile;
314
315     set myList {}
316     set myList2 {}
317
318     # Change path to the input file source to ensure relative paths work
319     set curDir [exec pwd];
320     cd [file dirname $inputFile]
321
322     for { set i 0 } { $i < [llength $inputFileLines] } {incr i 2} {
323         # check to see if file was added/removed
324
325         set norm1 [lindex $inputFileLines $i]
326         set norm1 [file normalize $norm1]
327
328         set norm2 [lindex $inputFileLines [expr $i+1]]
329         set norm2 [file normalize $norm2]
330
331         ## both norm1 and norm2 cant be empty
332         set norm1Empty [expr ![string length $norm1]]
333         set norm2Empty [expr ![string length $norm2]]
334
335         if { $norm1Empty && $norm2Empty } {

```



```

336     puts "Error with input file. Cannot have two blank lines $i and [expr $i+1]."
337     exit 1
338 }
339
340 if {!($norm1Empty)} {
341     set prevPathForLatex [escapePathForLatex $norm1]
342     set prevFileName [file tail $norm1]
343 }
344
345 if {!($norm2Empty)} {
346     set curPathForLatex [escapePathForLatex $norm2]
347     set curFileName [file tail $norm2]
348 }
349
350 # if added grab second line and insert to first that this was added
351 if {$norm1Empty} {
352     if { $verboseOutput } {
353         puts "added:\n$norm2"
354     }
355     # prev
356     addCodeToFile "$prevRoot/body/01-code.tex" "Added $curFileName" "New file: $curPathForLatex"
357         ↪ " "$norm2"
358     # cur
359     addCodeToFile "$curRoot/body/01-code.tex" "Added $curFileName" "New file: $curPathForLatex"
360         ↪ "$norm2"
361
362 # if removed grab first line and insert to next that this was removed
363 } elseif {$norm2Empty} {
364     if { $verboseOutput } {
365         puts "removed:\n$norm1"
366     }
367     # prev
368     addCodeToFile "$prevRoot/body/01-code.tex" "Removed $prevFileName" "Removed file:
369         ↪ $prevPathForLatex" "$norm1"
370     # cur
371     addCodeToFile "$curRoot/body/01-code.tex" "Removed $prevFileName" "Removed file:
372         ↪ $prevPathForLatex" "$norm1"
373
374 # if directory path doesn't equal then add change path
375 } elseif {[string compare [file dirname $norm1] [file dirname $norm2]] != 0} {
376     if { $verboseOutput } {
377         puts "path changed:\n$norm1\n$norm2"
378     }
379     # prev
380     addCodeToFile "$prevRoot/body/01-code.tex" "Changed Path $prevFileName" "Changed file path
381         ↪ from: $prevPathForLatex to $curPathForLatex" "$norm1"
382     # cur
383     addCodeToFile "$curRoot/body/01-code.tex" "Changed Path $curFileName" "Changed file path
384         ↪ from: $prevPathForLatex to $curPathForLatex" "$norm2"
385 } else {
386     # prev
387     addCodeToFile "$prevRoot/body/01-code.tex" "$prevFileName" "File path: $prevPathForLatex" "
388         ↪ $norm1"
389     # cur
390     addCodeToFile "$curRoot/body/01-code.tex" "$curFileName" "File path: $curPathForLatex" "
391         ↪ $norm2"
392 }
393 }

```

```

387     cd $curDir
388
389     if { $verboseOutput } {
390         puts "Finished looping through files..."
391     }
392
393     return;
394 }
395
396 processInputFile $inputFile $prevRoot $curRoot
397
398 proc inlineLatex {latexRoot} {
399     global verboseOutput
400     set prevDir [pwd]
401     set inlineMain "main2.tex"
402     set defaultMain "main.tex"
403     set defaultBib "references.bib"; #TODO still not sure if this needs to be a bbl or bib file
404     cd $latexRoot
405     exec latexexpand --keep-comments --expand-bbl $defaultBib $defaultMain > $inlineMain
406     if { $verboseOutput } {
407         puts "latexexpand --keep-comments --expand-bbl $defaultBib $defaultMain > $inlineMain"
408     }
409
410     cd $prevDir
411     return $inlineMain;
412 }
413
414 # inline latex to make it easier to do the diff
415 set inline [inlineLatex $prevRoot]
416 set inline2 [inlineLatex $curRoot]
417
418 if { $verboseOutput } {
419     puts "Finished inlining files..."
420 }
421
422 puts " Done."; # preparing latex files
423
424 # generate pdf
425 # uses latexdiff-vc run in the temp folder
426
427 cd "$tmpDir"
428
429 # this was the original diff latex, but produces latex output
430 # the following program latexdiff-vc is preferred because it creates direct to PDF
431 # manual diff to .tex
432 # exec latexdiff "$prevRoot/$inline" "$curRoot/$inline2" > "$tmpDir/main2-diff.tex"
433 # spawn latexdiff-vc --verbose --pdf "$prevRoot/$inline" "$curRoot/$inline2"
434
435 puts "If the following step hangs (more than 2 minutes) enter 'x' and hit enter."
436 puts -nonewline "Generating PDF..."
437 flush stdout
438
439 # TODO investigate if there will be security issues with input sources being malicious from TCL or
440     ↪ Latex
441 set cmd "latexdiff-vc --pdf \"$prevRoot/$inline\" \"$curRoot/$inline2\""
442 if { $verboseOutput } {
443     puts "Running command to generate PDF:"
444     puts $cmd
445 }

```

```
445
446 if {[catch {exec {*}$cmd} result]} {
447
448     # report errors and warnings,
449     # there shouldn't be any because we are generating everything here
450     # input code files shouldn't break anything
451     # todo change to expect script to allow interaction with above program in case something happens
452     # as of now the program halts if there is a latex warning that requiries user engagement
453     puts "\nInformation about error: $::errorInfo\n\n"
454
455     if {[file exists "$tmpDir/main2-diff.pdf"]} {
456         puts "Warning on diff generation!"
457         set exitCode 2
458     } else {
459         puts stderr "Error!"
460         set exitCode 1
461     }
462
463 } else {
464     set exitCode 0
465 }
466
467 puts " Done."; # generating PDF
468
469 if {[file exists "$tmpDir/main2-diff.pdf"]} {
470
471     exec mv "$tmpDir/main2-diff.pdf" $outputFile
472
473     if {[file exists "$outputFile"]} {
474         puts "PDF created at:\n$outputFile"
475         set exitCode 0
476     } else {
477         puts "Error generating output..."
478         set exitCode 1
479     }
480
481 } else {
482     puts "Error generating output..."
483     set exitCode 1
484 }
485
486 if { $keepTmpOutput } {
487     puts "Keeping temporary LaTeX output at:\n$tmpDir"
488 } else {
489     puts -nonewline "Removing temporary LaTeX output..."
490     flush stdout
491     exec rm -r $tmpDir
492     puts " Done."
493 }
494
495 exit $exitCode
```

3 Removed c file.c

Removed file: /home/betsalel/CygwinDocuments/code-diff-to-pdf/example/v1/c file.c

1 // Version 1 of file

```
2
3 #include <stdio.h>
4 int main( int argc, const char* argv[] )
5 {
6     // this is an added comment
7     printf( "I changed too!\n");
8 }
```

4 Changed Path tcl-file.tcl

Changed file path from: /home/betsalel/CygwinDocuments/code-diff-to-pdf/example/v1/folder with space at end/tcl-file.tcl to /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 puts "One last change..."
5 # example comment
6 %DIF >
7 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."
```

5 Changed Path js-file.js

Changed file path from: /home/betsalel/CygwinDocuments/code-diff-to-pdf/example/v1/folder\$with#Chars%;'!&/js-file.js to /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 :-)
```

6 Added sql-file.sql

New file: /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 );
```
