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

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.
#
# 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
# build and output the diff as a PDF
# exit code 0 success
# exit code 1 error, no PDF
# exit code 2 warnings, PDF generated, but there may be issues
# exit code 3 script terminated unexpectedly

# assumptions:
# 1. files will follow their extensions and the syntax highlighting will not care about RPL
# 2. the resultant pdf will have content in the format: title page, TOC, listings with diffs
# 3. the input file contains pairs of files with the first file being the previous version
# 4. blank lines in the input file means the file was added or removed
#
#
# to find files: find . -regex ".*[.][h]?[st]?[cjmq]?[csl]"
#
# features that would be nice to have:
# parse headings of file (i.e. license info) to remove common headings and display in introduction
# have a separate TOC of diffs
# list of acronymns / dictionary used in code
# rolling change list sorted by date
# support for comparisons for more than one version
# choosing changes by major, minor, or build numbers
# parsing out comments into preabmles
# combination with Arros system to append figures and descriptions connected with source code
# add user information about the changes
# compare with diffs and blame for Fossil, Git and other popular systems
# right now, this tool is a stepping stone towards generating a PDF version of the complete system
    ↪ modeled in Arros
# we want to see from version to version what source code is being changed without care for who did
    ↪  it
# in the future, when users are integrated into the system, it will be possible to concern
    ↪ ourselves with tracking who inputs what
# preamble in latex file
# don't include titles and latex things that aren't needed (bib, list of tables and figures)
# run Expect spawn on PDF generation to interact with Latex and output latexdiff command
#
#
# Common errors with latexdiff and latex:
# LaTeX Error: File 'ulem.sty' not found.
#  missing tex package -> install with 'tlmgr install PGK'
# WARNING: Inconsistency in length of input string and parsed string
#  Issue on Suse linux platform with texlive installed
#  latexdiff at older version. Update to version >= 1.3.
#   Download and install from https://www.ctan.org/tex-archive/support/latexdiff
# Error with 'src2pdf.tcl' as input due to puts $fp "\\end\{lstlisting\}"
#  Latex assumed end of listing when the '{'s were not escaped
#
# see https://blog.tcl.tk/1246 for Tcllib installation
package require Tcl 8.6
package require cmdline
package require fileutil

set exitCode 3;# default unexpected exit

set usage ": src2pdf.tcl \[-d\] -i source-list-file.txt \[-o path-to-output.pdf\] \[-t\] \[-v\]\
    ↪ noptions:"

set parameters {
  {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:"}
   {o.arg  "./output.pdf"  "Optional. Path and filename for result.\n\tDefault: "}
   {d                  "Use diff for input source list file."}
   {v               "Optional. Verbose output."}
   {t               "Optional. Keep temporary Latex output."}
}

if {[catch {array set options [cmdline::getoptions ::argv $parameters $usage]}]} {
    puts [cmdline::usage $parameters $usage]
    exit $exitCode
} else {

   if {  $options(v) } {
      puts "Verbose mode on."
      set verboseOutput 1
       parray options
   } else {
      set verboseOutput 0
   }

   if {  $options(t) } {
      puts "Keeping temporary LaTeX output."
      set keepTmpOutput 1
   } else {
      set keepTmpOutput 0
   }

   if {  $options(d) } {
      puts "Performing code src diff to pdf."
      set diffMode 1
   } else {
      puts "Performing regular src to pdf."
      set diffMode 0
   }

   if {[string length $options(o)] > 0} {
      set outputFile [file normalize $options(o)]
      puts "Output file set:\n$outputFile"
   } else {
      set outputFile [file normalize "./output.pdf"]
   }

   if {[string length $options(i)] > 0} {
      set inputFile [file normalize $options(i)]
      puts "Input file set:\n$inputFile"
   } else {
      puts "Missing source list file.\n\n"
       puts [cmdline::usage $parameters $usage]
      exit 1
   }
}

proc isLink { aFile } {
    return [expr ! [catch {file readlink ${aFile}}]]
}

if { ! ${tcl_interactive} } {
    ;# not being interactive is not enough!
    ;# we could be being included which makes us a script
    ;# in which case argv0 would not apply
    set ::myName [info script]
    ;#set myName ${argv0}
} else {
    set ::myName [info script]
}
if { [isLink ${::myName}] } {
    set ::myName [file readlink ${::myName}]
}
set ::myName [file normalize ${::myName}]
while {! [file isdirectory $::myName] && ! [catch {file readlink ${::myName}} newName] } {
    set ::myName ${newName}
}
if { ! [file isdirectory ${::myName}] } {
    set ::myBase [file dirname ${::myName}]
} else {
    set ::myBase ${::myName}
}

if { $verboseOutput } {
   puts "Base:\n${::myBase}"
}

proc printList {list} {
   puts "Total files: [llength $list]"

   foreach elem $list {
      if {[string length $elem] > 0} {
         puts "Opening $elem ..."
         set fp [open $elem r]
         set fileData [read $fp]
         puts $fileData
         close $fp
         puts "Closed $elem\n\n"
      }
   }

   return
}

puts -nonewline "Preparing LaTeX files...     "
flush stdout

proc prepTempDir {} {
   global ::myBase
   global diffMode

   ;# del prev, set magic string to avoid collision with existing dir for rm -rf
   ;# TODO redo magic number if there was a collision and the tmp dir with
   ;# that number already exists.
   set magicNumber  "latex-template-[expr {int(rand()*999999999999) + 1}]"
   set tmpDir      "tmp-$magicNumber"
   exec rm -rf $tmpDir ;# this is dangerous if $tmpDir is set to another directory
   exec mkdir -p $tmpDir

   set latexPrevDir  "$tmpDir/latex-prev/"
   set latexCurDir  "$tmpDir/latex-cur/"

   if { $diffMode } {
      ;# create latex and reference locations
      exec cp -R ${::myBase}/latex-template $latexPrevDir
      exec cp -R ${::myBase}/latex-template $latexCurDir
   } else {
      exec cp -R ${::myBase}/latex-template $tmpDir
   }

   return [list [file normalize $latexPrevDir] [file normalize $latexCurDir] [file normalize $tmpDir]];
}

set latexRoots  [prepTempDir]
set prevRoot  [lindex $latexRoots 0]
set curRoot   [lindex $latexRoots 1]
set tmpDir    [lindex $latexRoots 2]

proc cleanUp {keepTmpOutput} {
   global tmpDir

   if {[string length $tmpDir] > 0} {
      if { $keepTmpOutput } {
         puts "Keeping temporary LaTeX output at:\n$tmpDir"
      } else {
         puts -nonewline "Removing temporary LaTeX output..."
         flush stdout
         exec rm -r $tmpDir
         puts " Done."
      }
   }
}

proc escapePathForLatex {s} {
   global verboseOutput

   if { $verboseOutput } {
      puts "String before escaping: "
      puts $s
   }

   ;# special characters in latex must be escaped
    ;# \ { } & ^ $ % ;# _ ~

   regsub -all {([\\\{\}&\^\$%#_~])} $s {\\\1} s

   ;# add latex line breaks into paths
   regsub -all {([\\]?[\/&\^\$%#_~\-;'!])} $s {\1\\discretionary{}{}{}} s

   if { $verboseOutput } {
      puts "String after escaping: "
      puts $s
   }
   return $s
}

proc getLanguageFromFile {fileName} {
   global verboseOutput

   set fe [file extension $fileName]
   if { $verboseOutput } {
      puts "File extension: $fe"
   }

   set language ""
   switch $fe {
      .c {
         set language "C"
      }
      .sql {
         set language "SQL"
      }
      .js {
         set language "ECMAScript"
      }
      .tcl {
         set language "tcl"
      }
      default {}
   }

   return $language
}

proc addCodeToFile {texFilePath sectionText bodyText codeFilePath} {
   global verboseOutput

   set fp [open $texFilePath a+]
   puts $fp "\\section\{$sectionText\}"
   puts $fp "$bodyText"
   ;# puts $fp "\\label{sec:$captionText}"
   ;# label=code:$fileName
   ;# TODO ensure that the equivalent files have the same label to allow references to work
       ↪ correctly
   set fp1 [open $codeFilePath r]
   set fileData [read $fp1]
   close $fp1
   puts $fp "\\begin\{lstlisting\}\[language=[getLanguageFromFile $codeFilePath]\]"
   puts $fp $fileData
   puts $fp "\\end\{lstlisting\}"
   close $fp

   if { $verboseOutput } {
      puts "Wrote data into tex file..."
   }
   return;
}

proc processDiffInputFile {inputFile prevRoot curRoot} {
   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]
    set inputFileLines [split [read $fpInputFile] "\n"]
    if { $verboseOutput } {
       puts "File lines: [llength $inputFileLines]"
       puts "Input files $inputFileLines"
    }

    close $fpInputFile;

    ;# Change path to the input file source to ensure relative paths work
    set curDir [exec pwd];
    cd [file dirname $inputFile]

    for { set i 0} {$i < [llength $inputFileLines]} {incr i 2} {
       ;# check to see if file was added/removed

       set norm1 [lindex $inputFileLines $i]
       set norm1 [file normalize $norm1]

       set norm2 [lindex $inputFileLines [expr $i+1]]
       set norm2 [file normalize $norm2]

       ;## both norm1 and norm2 cant be empty
       set norm1Empty [expr ![string length $norm1]]
       set norm2Empty [expr ![string length $norm2]]

       if {$norm1Empty && $norm2Empty} {
          puts "Error with input file. Cannot have two blank lines $i and [expr $i+1]."
             cleanUp 1
          exit 1
       }

       if {!($norm1Empty)} {
          set prevPathForLatex [escapePathForLatex $norm1]
          set prevFileName [file tail $norm1]
       }

       if {!($norm2Empty)} {
          set curPathForLatex [escapePathForLatex $norm2]
          set curFileName [file tail $norm2]
       }

       ;# if added grab second line and insert to first that this was added
       if {$norm1Empty} {
          if { $verboseOutput } {
             puts "added:\n$norm2"
          }
          ;# prev
          addCodeToFile "$prevRoot/body/01-code.tex" "Added $curFileName" "New file: $curPathForLatex
              ↪ " "$norm2"
          ;# cur
          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]
    set inputFileLines [split [read $fpInputFile] "\n"]
    if { $verboseOutput } {
       puts "File lines: [llength $inputFileLines]"
       puts "Input files $inputFileLines"
    }

    close $fpInputFile;

    ;# Change path to the input file source to ensure relative paths work
    set curDir [exec pwd];
    cd [file dirname $inputFile]

    for { set i 0} {$i < [llength $inputFileLines]} {incr i} {
       ;# check to see if file was added/removed

       set norm1 [lindex $inputFileLines $i]
       set norm1 [file normalize $norm1]

       ;## both norm1 and norm2 cant be empty
       set norm1Empty [expr ![string length $norm1]]

       if {$norm1Empty} {
          puts "Error with input file. Cannot have blank line $i."
          exit 1
       }

       set pathForLatex [escapePathForLatex $norm1]
       set fileName [file tail $norm1]
       addCodeToFile "$tmpDir/latex-template/body/01-code.tex" "$fileName" "File path: $pathForLatex"
           ↪  "$norm1"
    }

    cd $curDir

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

    return;
 }

 if { $diffMode } {
    processDiffInputFile $inputFile $prevRoot $curRoot
 } else {
    processRegularInputFile $inputFile $tmpDir
 }

 proc inlineLatex {latexRoot} {
    global verboseOutput
    set prevDir [pwd]
    set inlineMain  "main2.tex"
    set defaultMain "main.tex"
    set defaultBib  "references.bib"; ;#TODO still not sure if this needs to be a bbl or bib file
    cd $latexRoot
    exec latexpand --keep-comments --expand-bbl $defaultBib $defaultMain > $inlineMain
    if {  $verboseOutput } {
       puts "latexpand --keep-comments --expand-bbl $defaultBib $defaultMain > $inlineMain"
    }

    cd $prevDir
    return $inlineMain;
 }

 if { $diffMode } {
    ;# inline latex to make it easier to do the diff
    set inline  [inlineLatex $prevRoot]
    set inline2 [inlineLatex $curRoot]
 } else {
    set inline  [inlineLatex "$tmpDir/latex-template"]
 }

 if {  $verboseOutput } {
    puts "Finished inlining files..."
 }

 puts " Done."; ;# preparing latex files

 # generate pdf
 # uses latexdiff-vc run in the temp folder

 cd "$tmpDir"

 # this was the original diff latex, but produces latex output
 # the following program latexdiff-vc is preferred because it creates direct to PDF
 # manual diff to .tex
 # exec latexdiff "$prevRoot/$inline" "$curRoot/$inline2" > "$tmpDir/main2-diff.tex"
 # spawn latexdiff-vc --verbose --pdf "$prevRoot/$inline" "$curRoot/$inline2"

 puts "If the following step hangs (more than 2 minutes) enter 'x' and hit enter."
 puts -nonewline "Generating PDF...            "
 flush stdout

 if { $diffMode } {
    ;# inline latex to make it easier to do the diff
    ;# TODO investigate if there will be security issues with input sources being malicious from TCL
         ↪  or Latex
    set cmd "latexdiff-vc --pdf \"$prevRoot/$inline\" \"$curRoot/$inline2\""
    set tmpOutputPdf "main2-diff.pdf"
 } else {
    set cmd "latexmk -pdf \"$tmpDir/latex-template/$inline\""
    set tmpOutputPdf "main2.pdf"
 }


 if { $verboseOutput } {
    puts "Running command to generate PDF:"
    puts $cmd
 }

 if {[catch {exec {*}$cmd} result]} {

    ;# report errors and warnings,
    ;# there shouldn't be any because we are generating everything here
    ;# input code files shouldn't break anything
    ;# todo change to expect script to allow interaction with above program in case something
         ↪ happens
    ;# as of now the program halts if there is a latex warning that requries user engagement
      puts "\nInformation about error: $::errorInfo\n\n"

     if {[file exists "$tmpDir/$tmpOutputPdf"]} {
      puts "Warning on diff generation!"
      set exitCode 2
     } else {
      puts stderr "Error!"
      set exitCode 1
     }

 } else {
     set exitCode 0
 }

 puts " Done."; ;# generating PDF

 if {[file exists "$tmpDir/$tmpOutputPdf"]} {

    exec mv "$tmpDir/$tmpOutputPdf" $outputFile

    if {[file exists "$outputFile"]} {
      puts "PDF created at:\n$outputFile"
      set exitCode 0
    } else {
      puts "Error generating output..."
      set exitCode 1
    }

 } else {
    puts "Error generating output..."
    set exitCode 1
 }

 cleanUp $keepTmpOutput

 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

```
#!/usr/bin/tclsh

puts "Hello World"
# example comment

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

```
console.log( ’Hello, world!’ );
// 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

```
DROP TABLE HELLO_WORLD;

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