TomPiler

Generated by Doxygen 1.9.3

1 TomPiler 1

1.0.1 Useful Pages 1

1.0.2 About 1

2 changelog 3

3 VSCode setup instructions 5

4 Tompiler Readme 7

4.1 Compiling 7

4.2 Using 7

4.3 Folder and file Descriptions 7

4.4 Included 3rd party library, CuTest. 8

4.5 Credits 8

5 Todo List 9

6 Data Structure Index 11

6.1 Data Structures 11

7 File Index 13

7.1 File List 13

8 Data Structure Documentation 15

8.1 TCompFiles Struct Reference 15

8.1.1 Detailed Description 15

8.1.2 Field Documentation 15

8.1.2.1 in 16

8.1.2.2 input_file_name 16

8.1.2.3 input_file_state 16

8.1.2.4 listing 16

8.1.2.5 listing_file_name 16

8.1.2.6 listing_file_state 16

8.1.2.7 out 16

8.1.2.8 output_file_name 16

8.1.2.9 output_file_state 17

8.1.2.10 temp 17

8.1.2.11 temp_file_name 17

8.1.2.12 terminate_requested 17

9 File Documentation 19

9.1 docs/changelog.md File Reference 19

9.2 docs/VSCode.md File Reference 19

9.3 Readme.md File Reference 19

9.4 src/compfiles.c File Reference 19

9.4.1 Function Documentation 19
9.4.1.1 CompFiles_CopyInputToOutputs() 20
9.4.1.2 CompFiles_DeInit() 20
9.4.1.3 CompFiles_GenerateTempFile() 20
9.4.1.4 CompFiles_Init() 20
9.4.1.5 CompFiles_LoadInputFile() 20
9.4.1.6 CompFiles_LoadListingFile() 21
9.4.1.7 CompFiles_LoadOutputFile() 21
9.4.1.8 CompFiles_LoadTempFile() 21
9.4.1.9 CompFiles_promptInputFilename() 22
9.4.1.10 CompFiles_promptOutputFilename() 22
9.4.1.11 CompFiles_promptUserOverwriteSelection() 23
9.4.1.12 CompFiles_ValidateFiles() 23
9.4.1.13 CompFiles_ValidateInputFile() 24
9.4.1.14 CompFiles_ValidateListingFile() 24
9.4.1.15 CompFiles_ValidateOutputFile() 24
9.5 src/compfiles.h File Reference 25
9.5.1 Detailed Description 26
9.5.2 Enumeration Type Documentation 26
9.5.2.1 COMPFILES_STATE 26
9.5.2.2 USER_OUTPUT_OVERWRITE_SELECTION 26
9.5.3 Function Documentation 27
9.5.3.1 CompFiles_CopyInputToOutputs() 27
9.5.3.2 CompFiles_DeInit() 27
9.5.3.3 CompFiles_GenerateTempFile() 28
9.5.3.4 CompFiles_Init() 28
9.5.3.5 CompFiles_LoadInputFile() 28
9.5.3.6 CompFiles_LoadListingFile() 28
9.5.3.7 CompFiles_LoadOutputFile() 29
9.5.3.8 CompFiles_LoadTempFile() 29
9.5.3.9 CompFiles_promptInputFilename() 29
9.5.3.10 CompFiles_promptOutputFilename() 30
9.5.3.11 CompFiles_promptUserOverwriteSelection() 30
9.5.3.12 CompFiles_ValidateFiles() 31
9.5.3.13 CompFiles_ValidateInputFile() 31
9.5.3.14 CompFiles_ValidateListingFile() 31
9.5.3.15 CompFiles_ValidateOutputFile() 32
9.5.4 Variable Documentation 32
9.5.4.1 CompFiles 32
9.6 compfiles.h 33
9.7 src/file_util.c File Reference 34
9.7.1 Function Documentation 34

9.7.1.1 addExtension() 34
9.7.1.2 backupFile() 35
9.7.1.3 fileExists() 35
9.7.1.4 filenameHasExtension() 36
9.7.1.5 getString() 37
9.7.1.6 removeExtension() 37
9.8 src/file_util.h File Reference 38
9.8.1 Detailed Description 39
9.8.2 Enumeration Type Documentation 39
9.8.2.1 FILE_EXISTS_ENUM 39
9.8.2.2 FILENAME_EXTENSION_PARSE 39
9.8.3 Function Documentation 39
9.8.3.1 addExtension() 39
9.8.3.2 backupFile() 40
9.8.3.3 fileExists() 40
9.8.3.4 filenameHasExtension() 41
9.8.3.5 getString() 42
9.8.3.6 removeExtension() 42
9.9 file_util.h 43
9.10 src/main.c File Reference 44
9.10.1 Detailed Description 44
9.10.2 Program 1 - fileopen 44
9.10.2.1 Group 3 44
9.10.3 Function Documentation 45
9.10.3.1 main() 45

Index **47**

Chapter 1

TomPiler

Version

0.2.5

1.0.1 Useful Pages

- compfiles.h
- file_util.h
- TCompFiles

1.0.2 About

Created by Group 3 for CSC-460, Language Translations with Dr. Pyzdrowski, at PennWest California.

Chapter 2

changelog

1/28/2023: Karl

- used doxygen to generate documentation

1/27/2023: Thomas, Karl

- wrote copy inputs to outputs function

1/26/2023 : All group members

- refactored file_util into two files: compfiles and file_util
- worked on logic for validating an output file name
- auto-generate temp file
- validate listing file in a similar way to output file
- combined validation functions into one validate func; just pass it the command line arguments

1/25/2023 : Thomas

- promptOutputFile()
- Modified getString() to use realloc

1/24/2023 : All group members

- worked on main logic
- changed CompFiles struct to be a state machine
- created promptFilename

1/23/2023 : Thomas and klm127

- changed Author comment to include e-mail and class name.

- removed old addExtension function, old promptFilename function, and closeFile function.
- added promptFilename and getString function(not yet covered by unit tests)
- removed all of the stdin swapping to a `separate repo`, and tested it, due to nagging bugs.
  - NOTE: It turned out that the bug was that `dup2` closes a file and `fclose` was being called afterwards.

moved test dependencies to a sub folder `lib` and updated compilation commands to use this on the include path

1/22/2023 : thomas and klm127

- added removeExtension function and tests
- confirmed getchar will read an 'enter'.
- thomas fixed prompting function to accept alternate inputs
- added backupFile function and tests
- Included tests for filepaths with directories
- redid filenameHasExtension. It now allows for filenames like ".bob" and doesnt allow filenames that end in slashes. It does allow folders to have '.'s in them.

1/21/2023 : klm127

- added #pragma region directives to header files. This is basically just markup for VSCode. Each of these regions can now be folded in Visual Studio or VSCode. This does not affect -ansi compilation on MinGW-↩W64 gcc; as far as I can tell. The purpose is to make the code much easier to navigate without relying on tab-based folding. `See Also: stackoverflow answer`
- Cleaned up comments, tab-based folding, etc.
- Fixed up the `addExtension` to use malloc to create a longer, concatenated string out of its inputs. Added unit tests for `addExtension`.
- Refactored std swapping test utility functions. The best way to test a prompter is now to use is to call `set↩STDin3`, get the value, then dont forget to call `restoreSTD3()` *before* making a test-based assertion.

1/20/2023 : All group members in collaboration

- created `promptUserOverwriteSelection`.
- created tests for `promptUserOverwriteSelection`. This was quite an involved task because we had to figure out how to temporarily replace stdin and stdout with alternative files so that we could test functionalities like `scanf`. Ultimately we were able to figure it out.

1/19/2023 : klm127

- changed directory structure, added docs, src, and tests
- created changelog, included CuTest's readme in the docs
- updated tasks.json in .vscode to configure code generation
- output file is now `fileopen.exe` due to interpretation of video instructions
- added .gitignore so we can exclude executables from github
- Added the testing suite CuTest. More info `here`
- Added the functions `fileExists` and `filenameHasExtension`
- Added unit tests for `fileExists` and `filenameHasExtension`

Chapter 3

VSCode setup instructions

VSCode provides a decent environment to work in C with its highly customizable features, low overhead, and rich extension options.

The folder `.vscode` configures the workspace for use with VSCode.

`tasks.json` describes build and run commands.

Ctrl+Shift+B will build and run the programs.

You may have to change compilerPath in c_cpp_properties.json to your own compiler.

I'm using GCC 8.1 (came with CodeBlocks) with the -ansi flag.

I referenced this article when setting up the VSCode environment. `Medium Article`

I referenced the `gcc documentation` while setting up the compiler.

Chapter 4

Tompiler Readme

Tompiler will be a relatively simple compiler built for educational and explorative purposes.

4.1 Compiling

Compiler configurations are stored in the .bat files. There are two of them.

- runTests.bat compiles and runs the tests.
- compile.bat compiles and runs the code.

4.2 Using

Running `compile.bat` will run the compiler after executing. You can also find the executable, `fileopen.exe`, in your `bin` directory.

It takes up to two command line arguments. The first argument can be an input file path while the second argument can be an output file path.

Place the `bin` directory on your system path if you want to be able to run tompiler from anywhere.

4.3 Folder and file Descriptions

- .vscode : Contains vscode configurations.
- docs : Contains additional documentation
- src : Contains source code
  - file_util.c / .h : file i/o for the compiler
  - main.c : Program entry point
- tests : Contains source code for tests
  - deps: Contains test dependencies
    - CuTest.c / .h : CuTest micro test framework
    - std_swapper.c / .h : For swapping stdin and out with files.
  - file_util_test.c / .h : tests for file util
  - main_test.c : entry point for test compilation

4.4 Included 3rd party library, CuTest.

`Link to Cutest page`

This is a small bit of code (only 340 lines!) that provides a unit testing skeleton.

4.5 Credits

- Tom Terhune, `ter1023@pennwest.edu`
- Karl Miller, `mil7865@pennnwest.edu`
- Anthony Stepich, `ste4864@pennwest.edu`

Chapter 5

Todo List

Global CompFiles_CopyInputToOutputs ()
    Not Covered by Unit Tests

Chapter 6

Data Structure Index

6.1 Data Structures

Here are the data structures with brief descriptions:

TCompFiles
Manages input and output files 15

Chapter 7

File Index

7.1 File List

Here is a list of all files with brief descriptions:

src/compfiles.c 19
src/compfiles.h
CompFiles struct and "methods" 25
src/file_util.c 34
src/file_util.h
Functions to assist with file operations 38
src/main.c
Program entry point 44

Chapter 8

Data Structure Documentation

8.1 TCompFiles Struct Reference

Manages input and output files.

```
#include <compfiles.h>
```

Data Fields

- FILE ∗ in
- FILE ∗ out
- FILE ∗ temp
- FILE ∗ listing
- short input_file_state
- short output_file_state
- short listing_file_state
- short terminate_requested
- char ∗ input_file_name
- char ∗ output_file_name
- char ∗ listing_file_name
- char ∗ temp_file_name

8.1.1 Detailed Description

Manages input and output files.

CompFiles is a globally accesible struct which maintains references to the loaded files.

It has a number of functions closely associated to it. In that way it is a class-like, but a singleton. There is only one CompFiles that ever should exist.

8.1.2 Field Documentation

8.1.2.1 in

```
FILE* in
```

A file pointer to an open input file.

8.1.2.2 input_file_name

```
char* input_file_name
```

The input filename.

8.1.2.3 input_file_state

```
short input_file_state
```

Determines the status of input file validation.

8.1.2.4 listing

```
FILE* listing
```

A file pointer to an open listing file.

8.1.2.5 listing_file_name

```
char* listing_file_name
```

The listing filename

8.1.2.6 listing_file_state

```
short listing_file_state
```

Determines the status of listing file validation.

8.1.2.7 out

```
FILE* out
```

A file pointer to an open output file.

8.1.2.8 output_file_name

```
char* output_file_name
```

The output filename,

8.1.2.9 output_file_state

```
short output_file_state
```

Determines the status of output file validation.

8.1.2.10 temp

```
FILE* temp
```

A file pointer to an open tmp file.

8.1.2.11 temp_file_name

```
char* temp_file_name
```

The temp filename

8.1.2.12 terminate_requested

```
short terminate_requested
```

1 indicates that a user requested to terminate the program.

The documentation for this struct was generated from the following file:

- src/compfiles.h

Chapter 9

File Documentation

9.1 docs/changelog.md File Reference

9.2 docs/VSCode.md File Reference

9.3 Readme.md File Reference

9.4 src/compfiles.c File Reference

```
#include "compfiles.h"
```

Functions

- void CompFiles_Init ()
- void CompFiles_GenerateTempFile ()
- void CompFiles_DeInit ()
- void CompFiles_LoadInputFile (FILE ∗newInputFile)
- void CompFiles_LoadOutputFile (FILE ∗newOutputFile)
- void CompFiles_LoadTempFile (FILE ∗newTempFile)
- void CompFiles_LoadListingFile (FILE ∗newListingFile)
- char ∗ CompFiles_promptInputFilename ()
- void CompFiles_CopyInputToOutputs ()
- short CompFiles_ValidateFiles (char ∗inputFilename, const char ∗outputFilename)
- short CompFiles_ValidateInputFile (char ∗filename)
- short CompFiles_ValidateOutputFile (const char ∗filename)
- short CompFiles_ValidateListingFile (const char ∗filename)
- char ∗ CompFiles_promptOutputFilename ()
- short CompFiles_promptUserOverwriteSelection ()

9.4.1 Function Documentation

9.4.1.1 CompFiles_CopyInputToOutputs()

```
void CompFiles_CopyInputToOutputs ( )
```

CompFiles_CopyInputToOutputs copies all the data from the input file to each of the output files. After execution, all output files (tmp, list, and out) will have text identical to the input files.

Warning

Precondition: All CompFiles file pointers must be open and ready to read/write.

Author

Thomas, Karl

Date

1/27/2023

Todo Not Covered by Unit Tests

9.4.1.2 CompFiles_DeInit()

```
void CompFiles_DeInit ( )
```

Closes any open files and returns CompFiles to the default values. Deletes the temp file.

9.4.1.3 CompFiles_GenerateTempFile()

```
void CompFiles_GenerateTempFile ( )
```

Generates a temporary file with a unique name. This file will be destroyed when CompFiles_DeInit() is called.

Author

klm127

Date

1/26/2023

9.4.1.4 CompFiles_Init()

```
void CompFiles_Init ( )
```

Initializes CompFiles struct to default values.

Note

Covered by unit tests.

9.4.1.5 CompFiles_LoadInputFile()

```
void CompFiles_LoadInputFile (
             FILE * newInputFile )
```

CompFiles_LoadInputFile loads a new file pointer as the input file. If there is a file already loaded, it closes that file first.

Parameters

newInputFile	A pointer to an open file in read mode.

9.4.1.6 CompFiles_LoadListingFile()

```
void CompFiles_LoadListingFile (
            FILE * newListingFile )
```

CompFiles_LoadListingFile loads a new file pointer as the listing file. If there is a file already loaded, it closes that file first.

Parameters

newOutputFile	A pointer to an open file in write mode.

9.4.1.7 CompFiles_LoadOutputFile()

```
void CompFiles_LoadOutputFile (
            FILE * newOutputFile )
```

CompFiles_LoadOutputFile loads a new file pointer as the output file. If there is a file already loaded, it closes that file first.

Parameters

newOutputFile	A pointer to an open file in write mode.

9.4.1.8 CompFiles_LoadTempFile()

```
void CompFiles_LoadTempFile (
            FILE * newTempFile )
```

CompFiles_LoadTempFile loads a new file pointer as the temp file. If there is a file already loaded, it closes that file first.

Parameters

newOutputFile	A pointer to an open file in write mode.

9.4.1.9 CompFiles_promptInputFilename()

```
char * CompFiles_promptInputFilename ( )
```

Calls the function getString() to recieve a filename from the user and returns it. It will set the 'terminate requested' flag in CompFiles if the user inputs only a \n.

Returns

char ∗ inputfilename to be verified

Author

thomaserh99

Date

1/23/2023

Note

Covered by Unit Tests

9.4.1.10 CompFiles_promptOutputFilename()

```
char * CompFiles_promptOutputFilename ( )
```

Calls the function getString() to recieve a filename from the user and returns it. It will set the 'terminate requested' flag in CompFiles if the user inputs only a \n.

Warning

This should not be called until the input filename has been set. The user may elect to generate an output filename based on the input file. (inputfilename + .out)

Returns

A malloced string of an output filename to be verified.

Author

thomaserh99

Date

Created On: 1/23/2023

Note

Covered by Unit Tests

9.4.1.11 CompFiles_promptUserOverwriteSelection()

```
short CompFiles_promptUserOverwriteSelection ( )
```

Prompts the user as to what they want to do about an output file already existing. It prints a prompt and parses the user response to one of the USER_OUTPUT_OVERWRITE_SELECTION enums. It does NOT loop.

Returns

short corresponding to one of the enums of USER_OTUPUT_OVERWRITE_SELECTION

Author

klm127, thomasterh99, anthony91501

Date

1/20/2023

Note

Covered by Unit Tests

9.4.1.12 CompFiles_ValidateFiles()

```
short CompFiles_ValidateFiles (
            char * inputFilename,
            const char * outputFilename )
```

Loops and prompts until all input and output files are set correctly or until terminate is requested.

Parameters

inputFilename	a filename with which to begin input validation with or NULL
outputFilename	a filename with which to begin output validation with or NULL

Returns

1 if terminate was requested. Otherwise, 0.

Author

klm127

Date

1/26/2023

9.4.1.13 CompFiles_ValidateInputFile()

```
short CompFiles_ValidateInputFile (
            char * filename )
```

Validates an input file name and sets the value in the struct. It will continue looping until the user has supplied a valid filename or elected to quit the program.

Parameters

filename	a filename with which to begin input validation with or NULL

Returns

0 if the input file was validated and loaded into the struct. 1 if the user requested to terminate the program.

9.4.1.14 CompFiles_ValidateListingFile()

```
short CompFiles_ValidateListingFile (
            const char * filename )
```

Validates a listing file name and sets the value in the struct.

Called by CompFiles_ValidateOutputFile after an output file has been fully validated. The parameter passed will be the name of the output file with the extension 'list' instead.

If this file happens to exist, a similar loop will occur as when a user attempts to load an extant output file. The user will be prompted to enter a new file until one is validated or they elect to exit the program.

Parameters

filename	a filename with which to begin input validation with or NULL

Returns

0 if an output file was validated and loaded into the struct. 1 if the user requested to terminate the program.

9.4.1.15 CompFiles_ValidateOutputFile()

```
short CompFiles_ValidateOutputFile (
            const char * filename )
```

Validates an output file name and sets the value in the struct. It will continue looping until the user has supplied a valid filename or elected to quit the program.

Parameters

filename	a filename with which to begin input validation with or NULL

Returns

0 if an output file was validated and loaded into the struct. 1 if the user requested to terminate the program.

9.5 src/compfiles.h File Reference

CompFiles struct and "methods".

```
#include <stdio.h>
#include "file_util.h"
#include <string.h>
#include <stdlib.h>
```

Data Structures

- struct TCompFiles

  Manages input and output files.

Enumerations

- enum COMPFILES_STATE { COMPFILES_STATE_NO_NAME_PROVIDED = 0 , COMPFILES_STATE_NAME_NEEDS_VALID
  = 1 , COMPFILES_STATE_NAME_VALIDATED = 2 }
- enum USER_OUTPUT_OVERWRITE_SELECTION {
  USER_OUTPUT_OVERWRITE_REENTER_FILENAME_SELECTED = 1 , USER_OUTPUT_OVERWRITE_OVERWRITE_EXI
  = 2 , USER_OUTPUT_OVERWRITE_DEFAULT_FILENAME = 3 , USER_OUTPUT_TERMINATE_PROGRAM
  = 4 ,
  USER_OUTPUT_TERMINATE_INVALID_ENTRY = -1 }

Functions

- void CompFiles_Init ()
- void CompFiles_DeInit ()
- void CompFiles_GenerateTempFile ()
- void CompFiles_LoadInputFile (FILE ∗newInputFile)
- void CompFiles_LoadOutputFile (FILE ∗newOutputFile)
- void CompFiles_LoadTempFile (FILE ∗newTempFile)
- void CompFiles_LoadListingFile (FILE ∗newListingFile)
- short CompFiles_ValidateFiles (char ∗inputFilename, const char ∗outputFilename)
- short CompFiles_ValidateInputFile (char ∗filename)
- short CompFiles_ValidateOutputFile (const char ∗filename)
- short CompFiles_ValidateListingFile (const char ∗filename)
- char ∗ CompFiles_promptInputFilename ()
- char ∗ CompFiles_promptOutputFilename ()
- short CompFiles_promptUserOverwriteSelection ()
- void CompFiles_CopyInputToOutputs ()

Variables

- struct TCompFiles CompFiles

9.5.1 Detailed Description

CompFiles struct and "methods".

CompFiles is a struct which holds pointers to the compilation input and output files. It also tracks their names and their validation status. It provides methods for prompting the user for valid file names until terminate is requested or all files are validated.

Authors

Tom Terhune, Karl Miller, Anthony Stepich

Date

January 2023

9.5.2 Enumeration Type Documentation

9.5.2.1 COMPFILES_STATE

```
enum COMPFILES_STATE
```

Describes the state of a filename validation process

Enumerator

COMPFILES_STATE_NO_NAME_PROVIDED	
COMPFILES_STATE_NAME_NEEDS_VALIDATION	
COMPFILES_STATE_NAME_VALIDATED	

9.5.2.2 USER_OUTPUT_OVERWRITE_SELECTION

```
enum USER_OUTPUT_OVERWRITE_SELECTION
```

Describes the possible selections a user may make when they elect to output to a file that already exists.

Enumerator

USER_OUTPUT_OVERWRITE_REENTER_FILENAME_SELECTED	

Enumerator

USER_OUTPUT_OVERWRITE_OVERWRITE_EXISTING_FILE	
USER_OUTPUT_OVERWRITE_DEFAULT_FILENAME	
USER_OUTPUT_TERMINATE_PROGRAM	
USER_OUTPUT_TERMINATE_INVALID_ENTRY	

9.5.3 Function Documentation

9.5.3.1 CompFiles_CopyInputToOutputs()

```
void CompFiles_CopyInputToOutputs ( )
```

CompFiles_CopyInputToOutputs copies all the data from the input file to each of the output files. After execution, all output files (tmp, list, and out) will have text identical to the input files.

Warning

Precondition: All CompFiles file pointers must be open and ready to read/write.

Author

Thomas, Karl

Date

1/27/2023

Todo Not Covered by Unit Tests

9.5.3.2 CompFiles_DeInit()

```
void CompFiles_DeInit ( )
```

Closes any open files and returns CompFiles to the default values. Deletes the temp file.

9.5.3.3 CompFiles_GenerateTempFile()

```
void CompFiles_GenerateTempFile ( )
```

Generates a temporary file with a unique name. This file will be destroyed when CompFiles_DeInit() is called.

Author

klm127

Date

1/26/2023

9.5.3.4 CompFiles_Init()

```
void CompFiles_Init ( )
```

Initializes CompFiles struct to default values.

Note

Covered by unit tests.

9.5.3.5 CompFiles_LoadInputFile()

```
void CompFiles_LoadInputFile (
            FILE * newInputFile )
```

CompFiles_LoadInputFile loads a new file pointer as the input file. If there is a file already loaded, it closes that file first.

Parameters

newInputFile	A pointer to an open file in read mode.

9.5.3.6 CompFiles_LoadListingFile()

```
void CompFiles_LoadListingFile (
            FILE * newListingFile )
```

CompFiles_LoadListingFile loads a new file pointer as the listing file. If there is a file already loaded, it closes that file first.

Parameters

newOutputFile	A pointer to an open file in write mode.

9.5.3.7 CompFiles_LoadOutputFile()

```
void CompFiles_LoadOutputFile (
            FILE * newOutputFile )
```

CompFiles_LoadOutputFile loads a new file pointer as the output file. If there is a file already loaded, it closes that file first.

Parameters

newOutputFile	A pointer to an open file in write mode.

9.5.3.8 CompFiles_LoadTempFile()

```
void CompFiles_LoadTempFile (
            FILE * newTempFile )
```

CompFiles_LoadTempFile loads a new file pointer as the temp file. If there is a file already loaded, it closes that file first.

Parameters

newOutputFile	A pointer to an open file in write mode.

9.5.3.9 CompFiles_promptInputFilename()

```
char * CompFiles_promptInputFilename ( )
```

Calls the function getString() to recieve a filename from the user and returns it. It will set the 'terminate requested' flag in CompFiles if the user inputs only a \n.

Returns

char ∗ inputfilename to be verified

Author

thomaserh99

Date

1/23/2023

Note

Covered by Unit Tests

9.5.3.10 CompFiles_promptOutputFilename()

```
char * CompFiles_promptOutputFilename ( )
```

Calls the function getString() to recieve a filename from the user and returns it. It will set the 'terminate requested' flag in CompFiles if the user inputs only a \n.

Warning

This should not be called until the input filename has been set. The user may elect to generate an output filename based on the input file. (inputfilename + .out)

Returns

A malloced string of an output filename to be verified.

Author

thomaserh99

Date

Created On: 1/23/2023

Note

Covered by Unit Tests

9.5.3.11 CompFiles_promptUserOverwriteSelection()

```
short CompFiles_promptUserOverwriteSelection ( )
```

Prompts the user as to what they want to do about an output file already existing. It prints a prompt and parses the user response to one of the USER_OUTPUT_OVERWRITE_SELECTION enums. It does NOT loop.

Returns

short corresponding to one of the enums of USER_OTUPUT_OVERWRITE_SELECTION

Author

klm127, thomasterh99, anthony91501

Date

1/20/2023

Note

Covered by Unit Tests

9.5.3.12 CompFiles_ValidateFiles()

```
short CompFiles_ValidateFiles (
            char * inputFilename,
            const char * outputFilename )
```

Loops and prompts until all input and output files are set correctly or until terminate is requested.

Parameters

inputFilename	a filename with which to begin input validation with or NULL
outputFilename	a filename with which to begin output validation with or NULL

Returns

1 if terminate was requested. Otherwise, 0.

Author

klm127

Date

1/26/2023

9.5.3.13 CompFiles_ValidateInputFile()

```
short CompFiles_ValidateInputFile (
            char * filename )
```

Validates an input file name and sets the value in the struct. It will continue looping until the user has supplied a valid filename or elected to quit the program.

Parameters

filename	a filename with which to begin input validation with or NULL

Returns

0 if the input file was validated and loaded into the struct. 1 if the user requested to terminate the program.

9.5.3.14 CompFiles_ValidateListingFile()

```
short CompFiles_ValidateListingFile (
            const char * filename )
```

Validates a listing file name and sets the value in the struct.

Called by CompFiles_ValidateOutputFile after an output file has been fully validated. The parameter passed will be the name of the output file with the extension 'list' instead.

If this file happens to exist, a similar loop will occur as when a user attempts to load an extant output file. The user will be prompted to enter a new file until one is validated or they elect to exit the program.

Parameters

filename	a filename with which to begin input validation with or NULL

Returns

0 if an output file was validated and loaded into the struct. 1 if the user requested to terminate the program.

9.5.3.15 CompFiles_ValidateOutputFile()

```
short CompFiles_ValidateOutputFile (
            const char * filename )
```

Validates an output file name and sets the value in the struct. It will continue looping until the user has supplied a valid filename or elected to quit the program.

Parameters

filename	a filename with which to begin input validation with or NULL

Returns

0 if an output file was validated and loaded into the struct. 1 if the user requested to terminate the program.

9.5.4 Variable Documentation

9.5.4.1 CompFiles

```
struct TCompFiles CompFiles
```

The CompFiles singleton.

9.6 compfiles.h

Go to the documentation of this file.

```

#ifndef compfiles_h
#define compfiles_h

#include <stdio.h>
#include "file_util.h"
#include <string.h>
#include <stdlib.h>

/*
-----------------
CompFies typedef
-----------------
*/
#pragma region structs
enum COMPFILES_STATE {
    COMPFILES_STATE_NO_NAME_PROVIDED = 0,
    COMPFILES_STATE_NAME_NEEDS_VALIDATION = 1,
    COMPFILES_STATE_NAME_VALIDATED = 2
};

struct TCompFiles {
    FILE * in;
    FILE * out;
    FILE * temp;
    FILE * listing;
    short input_file_state;
    short output_file_state;
    short listing_file_state;
    short terminate_requested;
    char * input_file_name;
    char * output_file_name;
    char * listing_file_name;
    char * temp_file_name;
};

struct TCompFiles CompFiles;

#pragma endregion structs

/*
-------------------
CompFiles lifecycle
-------------------
*/
#pragma region lifecycle

void CompFiles_Init();
void CompFiles_DeInit();
void CompFiles_GenerateTempFile();

#pragma endregion lifecycle

/*
-----------------
CompFiles setters
-----------------
*/
#pragma region setters

void CompFiles_LoadInputFile(FILE * newInputFile);

void CompFiles_LoadOutputFile(FILE * newOutputFile);

void CompFiles_LoadTempFile(FILE * newTempFile);

void CompFiles_LoadListingFile(FILE * newListingFile);

#pragma endregion setters

/*
------------------
CompFiles prompts
------------------
*/
#pragma region prompts

short CompFiles_ValidateFiles(char * inputFilename, const char * outputFilename);


short CompFiles_ValidateInputFile(char * filename);

short CompFiles_ValidateOutputFile(const char * filename);

short CompFiles_ValidateListingFile(const char * filename);

char * CompFiles_promptInputFilename();

char * CompFiles_promptOutputFilename();

enum USER_OUTPUT_OVERWRITE_SELECTION {
    USER_OUTPUT_OVERWRITE_REENTER_FILENAME_SELECTED = 1,
    USER_OUTPUT_OVERWRITE_OVERWRITE_EXISTING_FILE = 2,
    USER_OUTPUT_OVERWRITE_DEFAULT_FILENAME = 3,
    USER_OUTPUT_TERMINATE_PROGRAM = 4,
    USER_OUTPUT_TERMINATE_INVALID_ENTRY = -1
};
short CompFiles_promptUserOverwriteSelection();

#pragma endregion prompts

/*
--------------------
CompFiles operations
--------------------
*/
#pragma region operations

void CompFiles_CopyInputToOutputs();

#pragma endregion operations


#endif

```

9.7 src/file_util.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>
#include "file_util.h"
```

Functions

- short fileExists (const char ∗filename)
- void backupFile (const char ∗filename)
- int filenameHasExtension (const char ∗filename)
- char ∗ addExtension (const char ∗filename, const char ∗extension)
- char ∗ removeExtension (const char ∗filename)
- char ∗ getString ()

9.7.1 Function Documentation

9.7.1.1 addExtension()

```
char ∗ addExtension (
            const char ∗ filename,
            const char ∗ extension )
```

addExtension modifies the string given by filename by concatenating the string given by extension.

addExtension returns a pointer to a new, concatenated string. This string is allocated with `malloc`. When you are done with it, the memory should be cleared with `free` to avoid memory leaks.

Parameters

filename	the char array to modify
extension	the char array to append

Authors

thomasterh99, klm127

Date

1/18/2023

Note

Covered by Unit Tests

9.7.1.2 backupFile()

```
void backupFile (
             const char * filename )
```

Renames an existing file, adding the extension '.bak' to the end of it. For example 'outFile.out' will become 'out↩
File.out.bak'.

If the backup file exists, the function will recurse, backing up that file as well.

Author

klm127

Date

1/22/2023

Note

Covered by Unit Tests

9.7.1.3 fileExists()

```
short fileExists (
             const char * filename )
```

fileExists checks whether a file with name filename exists.

Parameters

filename	: the filename to check.

Returns

short:

- 1 if the file exists
- 0 if it does not.

Authors

klm127

Date

1/19/2023

Note

Covered by Unit Tests

9.7.1.4 filenameHasExtension()

```
int filenameHasExtension (
            const char * filename )
```

filenameHasExtension checks whether a filename has an extension. It validates that a string would be a valid path but with one additional condition: it must have a period in the file name portion of the path followed by at least one character.

Parameters

filename	the string to check

Returns

int:

- the index of the `.` character in the string if it exists. otherwise, one of the negative `FILE_EXTENSION↩ _PARSE` enums indicating why the filename is invalid;
  - (-1) means there was no period.
  - (-2) means it ended in a period.
  - (-3) means it is only a period.
  - (-4) means it ends in a slash and is a directory.

Author

klm127

Date

1/19/2023

Note

Covered by Unit Tests

9.7.1.5 getString()

```
char * getString ( )
```

getString scans a string character by character until recieving a null termination character or a new line

Returns

a pointer to a new character array given by the user with a size of the number of characters + 4 for the possible extension This string is allocated with `malloc`. When you are done with it, the memory should be cleared with `free` to avoid memory leaks.

Author

thomaserh99

Date

1/23/2023

Note

Covered by Unit Tests

9.7.1.6 removeExtension()

```
char * removeExtension (
            const char * filename )
```

removeExtension modifices the string given in parameters by copying the characters of the string up to the index of the last period.

Precondition

filename has been validated to have a correct extension (not leading with a '.', not ending with a '.')

Parameters

filename	the filename char∗ to remove the extension from.

Returns

a pointer to a new, extensionless string.

Warning

This string is allocated with `malloc`. When you are done with it, the memory should be cleared with `free` to avoid memory leaks.

Authors

thomasterh99, klm127

Date

1/22/2023

Note

Covered by Unit Tests

9.8 src/file_util.h File Reference

Functions to assist with file operations.

```
#include <stdbool.h>
#include <stdio.h>
```

Enumerations

- enum FILE_EXISTS_ENUM { FILE_EXISTS = 1 , FILE_DOES_NOT_EXIST = 0 }
- enum FILENAME_EXTENSION_PARSE { FILENAME_HAS_NO_PERIOD = -1 , FILENAME_ENDS_IN_PERIOD = -2 , FILENAME_IS_ONLY_PERIOD = -3 , FILENAME_IS_DIRECTORY = -4 }

Functions

- void backupFile (const char ∗filename)
- short fileExists (const char ∗filename)
- int filenameHasExtension (const char ∗filename)
- char ∗ addExtension (const char ∗filename, const char ∗extension)
- char ∗ removeExtension (const char ∗filename)
- char ∗ getString ()

9.8.1 Detailed Description

Functions to assist with file operations.

Authors

Karl Miller, Tom Terhune, Anthony Stepich

9.8.2 Enumeration Type Documentation

9.8.2.1 FILE_EXISTS_ENUM

```
enum FILE_EXISTS_ENUM
```

Alias for true false, 1, 0

Enumerator

FILE_EXISTS	
FILE_DOES_NOT_EXIST	

9.8.2.2 FILENAME_EXTENSION_PARSE

```
enum FILENAME_EXTENSION_PARSE
```

The enum FILENAME_EXTENSION_PARSE describes possible return values from filenameHasExtension which indicate different ways which a filename may be invalid.

Enumerator

FILENAME_HAS_NO_PERIOD	
FILENAME_ENDS_IN_PERIOD	
FILENAME_IS_ONLY_PERIOD	
FILENAME_IS_DIRECTORY	

9.8.3 Function Documentation

9.8.3.1 addExtension()

```
char * addExtension (
            const char * filename,
            const char * extension )
```

addExtension modifies the string given by filename by concatenating the string given by extension.

addExtension returns a pointer to a new, concatenated string. This string is allocated with `malloc`. When you are done with it, the memory should be cleared with `free` to avoid memory leaks.

Parameters

filename	the char array to modify
extension	the char array to append

Authors

thomasterh99, klm127

Date

1/18/2023

Note

Covered by Unit Tests

9.8.3.2 backupFile()

```
void backupFile (
            const char * filename )
```

Renames an existing file, adding the extension '.bak' to the end of it. For example 'outFile.out' will become 'out←File.out.bak'.

If the backup file exists, the function will recurse, backing up that file as well.

Author

klm127

Date

1/22/2023

Note

Covered by Unit Tests

9.8.3.3 fileExists()

```
short fileExists (
            const char * filename )
```

fileExists checks whether a file with name filename exists.

Parameters

filename	: the filename to check.

Returns

short:

- 1 if the file exists
- 0 if it does not.

Authors

klm127

Date

1/19/2023

Note

Covered by Unit Tests

9.8.3.4 filenameHasExtension()

```
int filenameHasExtension (
            const char * filename )
```

filenameHasExtension checks whether a filename has an extension. It validates that a string would be a valid path but with one additional condition: it must have a period in the file name portion of the path followed by at least one character.

Parameters

filename	the string to check

Returns

int:

- the index of the `.` character in the string if it exists. otherwise, one of the negative `FILE_EXTENSION↩ _PARSE` enums indicating why the filename is invalid;
  - (-1) means there was no period.
  - (-2) means it ended in a period.
  - (-3) means it is only a period.
  - (-4) means it ends in a slash and is a directory.

Author

klm127

Date

1/19/2023

Note

Covered by Unit Tests

9.8.3.5 getString()

```
char * getString ( )
```

getString scans a string character by character until recieving a null termination character or a new line

Returns

a pointer to a new character array given by the user with a size of the number of characters + 4 for the possible extension This string is allocated with `malloc`. When you are done with it, the memory should be cleared with `free` to avoid memory leaks.

Author

thomaserh99

Date

1/23/2023

Note

Covered by Unit Tests

9.8.3.6 removeExtension()

```
char * removeExtension (
            const char * filename )
```

removeExtension modifices the string given in parameters by copying the characters of the string up to the index of the last period.

Precondition

filename has been validated to have a correct extension (not leading with a '.', not ending with a '.')

Parameters

filename	the filename char∗ to remove the extension from.

Returns

a pointer to a new, extensionless string.

Warning

This string is allocated with `malloc`. When you are done with it, the memory should be cleared with `free` to avoid memory leaks.

Authors

thomasterh99, klm127

Date

1/22/2023

Note

Covered by Unit Tests

9.9 file_util.h

Go to the documentation of this file.

```
#ifndef file_util_h
#define file_util_h
#include <stdbool.h>
#include <stdio.h>

/*
----------------
file operations
----------------
*/
#pragma region fileops

void backupFile(const char * filename);

enum FILE_EXISTS_ENUM {
    FILE_EXISTS = 1,
    FILE_DOES_NOT_EXIST = 0
};
short fileExists(const char * filename);

#pragma endregion fileops

/*
-------------------
filename functions
-------------------
*/
#pragma region filenames


enum FILENAME_EXTENSION_PARSE {
    FILENAME_HAS_NO_PERIOD = -1,
    FILENAME_ENDS_IN_PERIOD = -2,
    FILENAME_IS_ONLY_PERIOD = -3,
    FILENAME_IS_DIRECTORY = -4
};

int filenameHasExtension(const char * filename);

char * addExtension(const char* filename, const char* extension);

char * removeExtension(const char * filename);



#pragma endregion filenames

/*
---------------------------
prompt assistance functions
---------------------------
*/
#pragma region prompts

char * getString();

#pragma endregion prompts


#endif

```

9.10 src/main.c File Reference

Program entry point.

```
#include "file_util.h"
#include "compfiles.h"
#include <stdio.h>
#include <string.h>
#include <stdbool.h>
#include <stdlib.h>
```

Functions

- int main (int argc, char ∗argv[ ])

9.10.1 Detailed Description

Program entry point.

Authors

Anthony Stepich
Tom Terhune
Karl Miller

9.10.2 Program 1 - fileopen

9.10.2.1 Group 3

9.10.2.1.1 CSC 460 - Language Translation

9.10.3 Function Documentation

9.10.3.1 main()

```
int main (
            int argc,
            char * argv[] )
```

Program entry point.

Index

addExtension
- file_util.c, 34
- file_util.h, 39

backupFile
- file_util.c, 35
- file_util.h, 40

CompFiles
- compfiles.h, 32

compfiles.c
- CompFiles_CopyInputToOutputs, 19
- CompFiles_DeInit, 20
- CompFiles_GenerateTempFile, 20
- CompFiles_Init, 20
- CompFiles_LoadInputFile, 20
- CompFiles_LoadListingFile, 21
- CompFiles_LoadOutputFile, 21
- CompFiles_LoadTempFile, 21
- CompFiles_promptInputFilename, 21
- CompFiles_promptOutputFilename, 22
- CompFiles_promptUserOverwriteSelection, 22
- CompFiles_ValidateFiles, 23
- CompFiles_ValidateInputFile, 23
- CompFiles_ValidateListingFile, 24
- CompFiles_ValidateOutputFile, 24

compfiles.h
- CompFiles, 32
- CompFiles_CopyInputToOutputs, 27
- CompFiles_DeInit, 27
- CompFiles_GenerateTempFile, 27
- CompFiles_Init, 28
- CompFiles_LoadInputFile, 28
- CompFiles_LoadListingFile, 28
- CompFiles_LoadOutputFile, 29
- CompFiles_LoadTempFile, 29
- CompFiles_promptInputFilename, 29
- CompFiles_promptOutputFilename, 30
- CompFiles_promptUserOverwriteSelection, 30
- COMPFILES_STATE, 26
- COMPFILES_STATE_NAME_NEEDS_VALIDATION, 26
- COMPFILES_STATE_NAME_VALIDATED, 26
- COMPFILES_STATE_NO_NAME_PROVIDED, 26
- CompFiles_ValidateFiles, 30
- CompFiles_ValidateInputFile, 31
- CompFiles_ValidateListingFile, 31
- CompFiles_ValidateOutputFile, 32
- USER_OUTPUT_OVERWRITE_DEFAULT_FILENAME, 27
- USER_OUTPUT_OVERWRITE_OVERWRITE_EXISTING_FILE, 27
- USER_OUTPUT_OVERWRITE_REENTER_FILENAME_SELECTED 26
- USER_OUTPUT_OVERWRITE_SELECTION, 26
- USER_OUTPUT_TERMINATE_INVALID_ENTRY, 27
- USER_OUTPUT_TERMINATE_PROGRAM, 27

CompFiles_CopyInputToOutputs
- compfiles.c, 19
- compfiles.h, 27

CompFiles_DeInit
- compfiles.c, 20
- compfiles.h, 27

CompFiles_GenerateTempFile
- compfiles.c, 20
- compfiles.h, 27

CompFiles_Init
- compfiles.c, 20
- compfiles.h, 28

CompFiles_LoadInputFile
- compfiles.c, 20
- compfiles.h, 28

CompFiles_LoadListingFile
- compfiles.c, 21
- compfiles.h, 28

CompFiles_LoadOutputFile
- compfiles.c, 21
- compfiles.h, 29

CompFiles_LoadTempFile
- compfiles.c, 21
- compfiles.h, 29

CompFiles_promptInputFilename
- compfiles.c, 21
- compfiles.h, 29

CompFiles_promptOutputFilename
- compfiles.c, 22
- compfiles.h, 30

CompFiles_promptUserOverwriteSelection
- compfiles.c, 22
- compfiles.h, 30

COMPFILES_STATE
- compfiles.h, 26

COMPFILES_STATE_NAME_NEEDS_VALIDATION
- compfiles.h, 26

COMPFILES_STATE_NAME_VALIDATED
- compfiles.h, 26

COMPFILES_STATE_NO_NAME_PROVIDED
- compfiles.h, 26

CompFiles_ValidateFiles
compfiles.c, 23
compfiles.h, 30
CompFiles_ValidateInputFile
compfiles.c, 23
compfiles.h, 31
CompFiles_ValidateListingFile
compfiles.c, 24
compfiles.h, 31
CompFiles_ValidateOutputFile
compfiles.c, 24
compfiles.h, 32

docs/changelog.md, 19
docs/VSCode.md, 19

FILE_DOES_NOT_EXIST
file_util.h, 39
FILE_EXISTS
file_util.h, 39
FILE_EXISTS_ENUM
file_util.h, 39
file_util.c
addExtension, 34
backupFile, 35
fileExists, 35
filenameHasExtension, 36
getString, 37
removeExtension, 37
file_util.h
addExtension, 39
backupFile, 40
FILE_DOES_NOT_EXIST, 39
FILE_EXISTS, 39
FILE_EXISTS_ENUM, 39
fileExists, 40
FILENAME_ENDS_IN_PERIOD, 39
FILENAME_EXTENSION_PARSE, 39
FILENAME_HAS_NO_PERIOD, 39
FILENAME_IS_DIRECTORY, 39
FILENAME_IS_ONLY_PERIOD, 39
filenameHasExtension, 41
getString, 42
removeExtension, 42
fileExists
file_util.c, 35
file_util.h, 40
FILENAME_ENDS_IN_PERIOD
file_util.h, 39
FILENAME_EXTENSION_PARSE
file_util.h, 39
FILENAME_HAS_NO_PERIOD
file_util.h, 39
FILENAME_IS_DIRECTORY
file_util.h, 39
FILENAME_IS_ONLY_PERIOD
file_util.h, 39
filenameHasExtension
file_util.c, 36
file_util.h, 41

getString
file_util.c, 37
file_util.h, 42

in
TCompFiles, 15
input_file_name
TCompFiles, 16
input_file_state
TCompFiles, 16

listing
TCompFiles, 16
listing_file_name
TCompFiles, 16
listing_file_state
TCompFiles, 16

main
main.c, 45
main.c
main, 45

out
TCompFiles, 16
output_file_name
TCompFiles, 16
output_file_state
TCompFiles, 16

Readme.md, 19
removeExtension
file_util.c, 37
file_util.h, 42

src/compfiles.c, 19
src/compfiles.h, 25, 33
src/file_util.c, 34
src/file_util.h, 38, 43
src/main.c, 44

TCompFiles, 15
in, 15
input_file_name, 16
input_file_state, 16
listing, 16
listing_file_name, 16
listing_file_state, 16
out, 16
output_file_name, 16
output_file_state, 16
temp, 17
temp_file_name, 17
terminate_requested, 17
temp
TCompFiles, 17
temp_file_name
TCompFiles, 17

terminate_requested
TCompFiles, 17

USER_OUTPUT_OVERWRITE_DEFAULT_FILENAME
compfiles.h, 27
USER_OUTPUT_OVERWRITE_OVERWRITE_EXISTING_FILE
compfiles.h, 27
USER_OUTPUT_OVERWRITE_REENTER_FILENAME_SELECTED
compfiles.h, 26
USER_OUTPUT_OVERWRITE_SELECTION
compfiles.h, 26
USER_OUTPUT_TERMINATE_INVALID_ENTRY
compfiles.h, 27
USER_OUTPUT_TERMINATE_PROGRAM
compfiles.h, 27