

scaf

Generated by Doxygen 1.9.3

1	scaf	1
2	Deprecated List	3
3	Namespace Index	5
3.1	Namespace List	5
4	Class Index	7
4.1	Class List	7
5	File Index	9
5.1	File List	9
6	Namespace Documentation	11
6.1	Filer Namespace Reference	11
6.1.1	Detailed Description	11
6.1.2	Function Documentation	11
6.1.2.1	clearDir()	11
6.1.2.2	copyRecursive()	12
6.1.2.3	fillMapWithDirectories()	12
6.1.2.4	fillVectorWithDirectories() [1/2]	13
6.1.2.5	fillVectorWithDirectories() [2/2]	13
6.1.2.6	isEmpty()	14
7	Class Documentation	15
7.1	Config Class Reference	15
7.1.1	Detailed Description	15
7.1.2	Constructor & Destructor Documentation	15
7.1.2.1	Config() [1/2]	16
7.1.2.2	Config() [2/2]	16
7.1.3	Member Function Documentation	16
7.1.3.1	getInfo()	16
7.1.3.2	getPath()	16
7.1.3.3	getTemplateDir()	16
7.1.3.4	hasTemplateDir()	16
7.1.3.5	readConfig()	17
7.1.3.6	setInfo()	17
7.1.3.7	setTemplateDir()	17
7.1.3.8	writeConfig()	17
7.2	ConfigTest Class Reference	18
7.2.1	Detailed Description	18
7.3	Filer::copy_result Struct Reference	18
7.3.1	Detailed Description	18
7.3.2	Member Data Documentation	18

7.3.2.1 filescopied	19
7.3.2.2 folderscopied	19
7.3.2.3 gitignored	19
7.4 FilerTest Class Reference	19
7.4.1 Detailed Description	19
7.5 Scaf Class Reference	19
7.5.1 Detailed Description	20
7.5.2 Constructor & Destructor Documentation	20
7.5.2.1 Scaf() [1/2]	20
7.5.2.2 Scaf() [2/2]	20
7.5.3 Member Function Documentation	20
7.5.3.1 Start()	20
7.6 Template Class Reference	21
7.6.1 Detailed Description	21
7.6.2 Member Function Documentation	21
7.6.2.1 getAlias()	22
7.6.2.2 getInfo()	22
7.6.2.3 getPath()	22
7.6.2.4 setAlias()	22
7.6.2.5 setInfo()	22
7.7 TemplateTest Class Reference	22
7.7.1 Detailed Description	22
8 File Documentation	23
8.1 src/Config.cpp File Reference	23
8.1.1 Detailed Description	23
8.1.2 Function Documentation	23
8.1.2.1 GetFullExePath()	24
8.2 src/Config.h File Reference	24
8.2.1 Detailed Description	24
8.2.2 Function Documentation	25
8.2.2.1 GetFullExePath()	25
8.2.3 Variable Documentation	25
8.2.3.1 dirKey	25
8.2.3.2 infoKey	25
8.3 Config.h	26
8.4 src/Filer.cpp File Reference	26
8.4.1 Detailed Description	26
8.5 src/Filer.h File Reference	27
8.5.1 Detailed Description	27
8.6 Filer.h	28
8.7 src/Scaf.h File Reference	28

8.7.1 Detailed Description	28
8.7.2 Function Documentation	29
8.7.2.1 printCopyResult()	29
8.7.2.2 promptYN()	29
8.7.2.3 stringLower()	29
8.8 Scaf.h	29
8.9 src/Template.cpp File Reference	30
8.9.1 Detailed Description	30
8.10 src/Template.h File Reference	31
8.10.1 Detailed Description	31
8.11 Template.h	31
8.12 tests/ConfigTest.cpp File Reference	32
8.12.1 Detailed Description	32
8.13 tests/FilerTest.cpp File Reference	32
8.13.1 Detailed Description	33
8.14 tests/TemplateTest.cpp File Reference	33
8.14.1 Detailed Description	33
8.15 tests/testAll.cpp File Reference	34
8.15.1 Detailed Description	34
8.15.2 Function Documentation	34
8.15.2.1 main()	34
Index	35

Chapter 1

scaf

Version

1.0.0

Created by Karl Miller for the Spring 2023 Code Jam at PennWest California.

[Scaf](#) is general purpose, command line, project initialization tool.

[Scaf](#) is purpose-agnostic and can be used to start-up ("scaffold") any type of project that has a directory structure.

[Scaf](#) works by maintaining directories of templates that the user supplies. When the user wants to scaffold a new project, they may use `scaf` to copy the contents from one of these directories into their current directory.

Chapter 2

Deprecated List

File [Template.cpp](#)

File [Template.h](#)

File [TemplateTest.cpp](#)

Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Filer	Performs some filesystem operations and queries	11
-----------------------	---	--------------------

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Config	Loads scaf.conf.json from the process working directory	15
ConfigTest	Tests the Config class	18
Filer::copy_result	Provides data about a recursive copy operation	18
FilerTest	Tests the Filer namespace	19
Scaf	Parses and executes the command line arguments	19
Template	Holds information about a template. Not used!	21
TemplateTest	Tests the Template class	22

Chapter 5

File Index

5.1 File List

Here is a list of all documented files with brief descriptions:

src/ Config.cpp	
Implementation for Config class	23
src/ Config.h	
Declarations for Config class	24
src/ Filer.cpp	
Definitions for Filer namespace	26
src/ Filer.h	
Declarations for Filer namespace	27
src/ Scaf.h	
Declarations for Scaf class	28
src/ Template.cpp	
Definitions for Template class (deprecated)	30
src/ Template.h	
Declarations for Template class	31
tests/ ConfigTest.cpp	
Provides tests for Config . See Config.h	32
tests/ FilerTest.cpp	
Provides tests for Filer . See Filer.h	32
tests/ TemplateTest.cpp	
Provides tests for Template . See Template.h	33
tests/ testAll.cpp	
Calls appropriate unit tests by parsing command line arguments	34

Chapter 6

Namespace Documentation

6.1 Filer Namespace Reference

Performs some filesystem operations and queries.

Classes

- struct [copy_result](#)
Provides data about a recursive copy operation.

Functions

- [copy_result copyRecursive](#) (fs::path from, fs::path to)
- bool [isEmpty](#) (fs::path checkdir)
- bool [clearDir](#) (fs::path dirToClear)
- bool [fillMapWithDirectories](#) (const fs::path &p, map< string, fs::path > &m)
- bool [fillVectorWithDirectories](#) (const fs::path &p, vector< string > &m)
- bool [fillVectorWithDirectories](#) (const fs::path &p, vector< string > &v, string &filter_prefix)

6.1.1 Detailed Description

Performs some filesystem operations and queries.

[Filer](#) provides functions wrapping various filesystem operations, such as [copyRecursive](#) and [isEmpty](#), used by [Scaf](#).

Particularly, [Filer](#) ignores `.git` folders when determining whether a directory is empty and when copying recursively.

It also provides functions for getting information about the contents of a folder.

6.1.2 Function Documentation

6.1.2.1 `clearDir()`

```
bool Filer::clearDir (
    fs::path dirToClear )
```

Clears a directory of all its contents, except for `.git` folder.

Parameters

<i>dirToClear</i>	The directory to clear of contents.
-------------------	-------------------------------------

Returns

true if path was valid and not a directory, false otherwise.

6.1.2.2 copyRecursive()

```
Filer::copy_result Filer::copyRecursive (
    fs::path from,
    fs::path to )
```

Recursively copies all files and folders in from to directory to.

Excludes .git folder and subdirs.

Precondition

from and to must be existing directories.

Parameters

<i>from</i>	The directory to copy from.
<i>to</i>	The directory to copy to.

Returns

The number of files copied.

6.1.2.3 fillMapWithDirectories()

```
bool Filer::fillMapWithDirectories (
    const fs::path & p,
    map< string, fs::path > & m )
```

Fills a map in-place with directory names inside of path.

Parameters

<i>p</i>	The path to use to populate the map.
<i>m</i>	The map to fill in-place.

Returns

true if path as valid and not a directory, false otherwise.

6.1.2.4 fillVectorWithDirectories() [1/2]

```
bool Filer::fillVectorWithDirectories (
    const fs::path & p,
    vector< string > & m )
```

Fills a vector in-place with directory names inside of path, then sorts the vector.

Parameters

<i>p</i>	The path to use to fill the vector.
<i>v</i>	The vector to fill in-place.

Returns

true if path was valid and not a directory, false otherwise.

6.1.2.5 fillVectorWithDirectories() [2/2]

```
bool Filer::fillVectorWithDirectories (
    const fs::path & p,
    vector< string > & v,
    string & filter_prefix )
```

Fills a vector in-place with directory names inside of path, then sorts the vector. Overload allows passing a string function to prefix-filter the names.

Parameters

<i>p</i>	The path to use to fill the vector.
<i>v</i>	The vector to fill in-place.
<i>filter_prefix</i>	A function to test each directory name. Directory names will only be added to v if callback(dirname) == true.

Returns

true if path was valid and not a directory, false otherwise.

6.1.2.6 isEmpty()

```
bool Filer::isEmpty (
    fs::path checkdir )
```

Checks whether a given directory is empty. (Excluding .git folder.)

Parameters

<i>checkdir</i>	The directory to examine.
-----------------	---------------------------

Returns

Whether the given directory is empty.

Chapter 7

Class Documentation

7.1 Config Class Reference

loads `scaf.conf.json` from the process working directory.

```
#include <Config.h>
```

Public Member Functions

- [Config](#) ()
- [Config](#) (fs::path customPath)
- fs::path [getPath](#) ()
- void [readConfig](#) ()
- fs::path [getTemplateDir](#) ()
- bool [setTemplateDir](#) (fs::path newdir)
- bool [hasTemplateDir](#) ()
- string [getInfo](#) (string template_name)
- void [setInfo](#) (string key, string value)
- void [writeConfig](#) ()

7.1.1 Detailed Description

loads `scaf.conf.json` from the process working directory.

[Config](#) provides the functions and classes needed to load `scaf`'s configuration.

When it is constructed, it looks for `scaf.config.json` in the process working directory. If it doesn't exist, it is created. If it exists as a directory, a runtime error is thrown.

Otherwise, it parses that configuration file to load `scaf`'s saved settings.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 Config() [1/2]

```
Config::Config ( )
```

Initializes the [Config](#). Looks for `scaf.config.json` in the process working directory. If it doesn't exist, it is created. If it exists as a directory, a runtime error is thrown.

7.1.2.2 Config() [2/2]

```
Config::Config (
    fs::path customPath )
```

Generally the 0 parameter constructor should be called. Passing a custom configuration is useful for testing purposes.

Parameters

<i>customPath</i>	The path to configuration file.
-------------------	---------------------------------

7.1.3 Member Function Documentation

7.1.3.1 getInfo()

```
string Config::getInfo (
    string template_name )
```

Gets the info for a given template. Returns empty string if no such info exists.

7.1.3.2 getPath()

```
fs::path Config::getPath ( )
```

Returns the path that was loaded at the time of [Config](#)'s construction.

7.1.3.3 getTemplateDir()

```
fs::path Config::getTemplateDir ( )
```

Returns the template directory read from the config file.

7.1.3.4 hasTemplateDir()

```
bool Config::hasTemplateDir ( )
```

Checks if [Config](#) has a template directory loaded.

7.1.3.5 readConfig()

```
void Config::readConfig ( )
```

Uses json library to read the configuration file.

If the json object is misconfigured, it may simply skip parsing that step and print an error to the console.

7.1.3.6 setInfo()

```
void Config::setInfo (
    string key,
    string value )
```

Sets the info for a given template.

Parameters

<i>key</i>	The name of the template.
<i>value</i>	The new info to set.

7.1.3.7 setTemplateDir()

```
bool Config::setTemplateDir (
    fs::path newdir )
```

Sets a new template directory. Will not be set and an error will be printed if the directory doesn't exist. If the directory is relative, it will be converted to absolute.

Parameters

<i>newdir</i>	The new template directory.
---------------	-----------------------------

Returns

true if directory could be set. false otherwise.

7.1.3.8 writeConfig()

```
void Config::writeConfig ( )
```

Writes the config as a json. Called on program end to record updates in the config file. Pretty-prints the JSON with indentation.

The documentation for this class was generated from the following files:

- [src/Config.h](#)
- [src/Config.cpp](#)

7.2 ConfigTest Class Reference

Tests the [Config](#) class.

Public Member Functions

- **ConfigTest** (std::ostream &out, int verbose_level=QUnit::verbose)
- int **run** ()

7.2.1 Detailed Description

Tests the [Config](#) class.

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

- tests/[ConfigTest.cpp](#)

7.3 Filer::copy_result Struct Reference

Provides data about a recursive copy operation.

```
#include <Filer.h>
```

Public Attributes

- int [filescopied](#)
- int [folderscopied](#)
- bool [gitsskipped](#)

7.3.1 Detailed Description

Provides data about a recursive copy operation.

Provides data about a copy operation.

Returned by [Filer::copyRecursive](#).

7.3.2 Member Data Documentation

7.3.2.1 filescopied

```
int Filer::copy_result::filescopied
```

The number of files copied.

7.3.2.2 folderscopied

```
int Filer::copy_result::folderscopied
```

The number of folders copied.

7.3.2.3 gitsskipped

```
bool Filer::copy_result::gitsskipped
```

Whether a '.git' folder was skipped.

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

- [src/Filer.h](#)

7.4 FilerTest Class Reference

Tests the [Filer](#) namespace.

Public Member Functions

- **FilerTest** (std::ostream &out, int verbose_level=QUnit::verbose)
- int **run** ()

7.4.1 Detailed Description

Tests the [Filer](#) namespace.

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

- [tests/FilerTest.cpp](#)

7.5 Scaf Class Reference

Parses and executes the command line arguments.

```
#include <Scaf.h>
```

Public Member Functions

- [Scaf](#) ()
- [Scaf](#) (filesystem::path config_path)
- bool [Start](#) (int argc, char **argv)

7.5.1 Detailed Description

Parses and executes the command line arguments.

[Scaf](#) drives the program execution. It loads the config and parses the command line arguments into the correct commands.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 [Scaf\(\)](#) [1/2]

```
Scaf::Scaf ( )
```

Constructor.

7.5.2.2 [Scaf\(\)](#) [2/2]

```
Scaf::Scaf (
    filesystem::path config_path )
```

Constructor with config override. For use with testing.

Parameters

<i>config_path</i>	A different configuration path to provide to the Config object.
--------------------	---

7.5.3 Member Function Documentation

7.5.3.1 [Start\(\)](#)

```
bool Scaf::Start (
    int argc,
    char ** argv )
```

Begins the parse process with the command-line args.

Parameters

<i>argc</i>	The arg count.
<i>argv</i>	The arg values.

Returns

True for succesful parse. False if there were errors.

The documentation for this class was generated from the following files:

- [src/Scaf.h](#)
- [src/Scaf.cpp](#)

7.6 Template Class Reference

Holds information about a template. Not used!

```
#include <Template.h>
```

Public Member Functions

- **Template** (string a_path, string an_alias, string some_info)
- **Template** (string a_path, string an_alias)
- **Template** (string a_path)
- string [getPath](#) ()
- string [getAlias](#) ()
- void [setAlias](#) (string an_alias)
- string [getInfo](#) ()
- void [setInfo](#) (string some_info)

7.6.1 Detailed Description

Holds information about a template. Not used!

[Template](#) holds information about a given template, including its folder location and saved-user info.

NOTE: TEMPLATE IS NOT CURRENTLY ACTUALLY USED IN THE PROGRAM!

Instead of instancing templates, scaf scans the directories within the root folder to get template names instead! As [Config](#) reads the saved program data, it instances Templates. (It doesn't!)

[Template](#) has not been removed because it is well-tested and may be useful in a future refactor.

7.6.2 Member Function Documentation

7.6.2.1 `getAlias()`

```
string Template::getAlias ( )
```

Gets the template's name.

7.6.2.2 `getInfo()`

```
string Template::getInfo ( )
```

If info is an empty string, returns a message saying that there is no info.

7.6.2.3 `getPath()`

```
string Template::getPath ( )
```

Gets the template's absolute path on the file system.

7.6.2.4 `setAlias()`

```
void Template::setAlias (
    string an_alias )
```

Sets the alias.

7.6.2.5 `setInfo()`

```
void Template::setInfo (
    string some_info )
```

Sets the template's info.

The documentation for this class was generated from the following files:

- [src/Template.h](#)
- [src/Template.cpp](#)

7.7 TemplateTest Class Reference

Tests the [Template](#) class.

Public Member Functions

- **TemplateTest** (std::ostream &out, int verbose_level=QUnit::verbose)
- int **run** ()

7.7.1 Detailed Description

Tests the [Template](#) class.

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

- [tests/TemplateTest.cpp](#)

Chapter 8

File Documentation

8.1 src/Config.cpp File Reference

Implementation for [Config](#) class.

```
#include <filesystem>
#include <fstream>
#include <Config.h>
#include <json.hpp>
```

Typedefs

- using **json** = nlohmann::json

Functions

- fs::path [GetFullExePath](#) ()

8.1.1 Detailed Description

Implementation for [Config](#) class.

Author

Karl Miller

Date

April 2023

8.1.2 Function Documentation

8.1.2.1 GetFullExePath()

```
fs::path GetFullExePath ( )
```

GetFullExePath is used to get the actual fs location of scaf. This is different from where the current working directory is. It's often referred to as the process working directory. Apparently pwd functions are not cross-platform in the built-in libraries, so directives are used to make scaf compatible on linux and windows.

This is used to locate scaf's config in the same directory as the scaf executable.

Returns

A path for the folder containing scaf.exe

From <https://stackoverflow.com/questions/50889647/best-way-to-get-exe-folder-path>

8.2 src/Config.h File Reference

Declarations for [Config](#) class.

```
#include <string>
#include <filesystem>
#include <iostream>
#include <map>
```

Classes

- class [Config](#)
loads scaf.conf.json from the process working directory.

Functions

- fs::path [GetFullExePath](#) ()

Variables

- const string [dirKey](#) = "templateDir"
- const string [infoKey](#) = "infos"

8.2.1 Detailed Description

Declarations for [Config](#) class.

Author

Karl Miller

Date

April 2023

8.2.2 Function Documentation

8.2.2.1 GetFullExePath()

```
fs::path GetFullExePath ( )
```

GetFullExePath is used to get the actual fs location of scaf. This is different from where the current working directory is. It's often referred to as the process working directory. Apparently pwd functions are not cross-platform in the built-in libraries, so directives are used to make scaf compatible on linux and windows.

This is used to locate scaf's config in the same directory as the scaf executable.

Returns

A path for the folder containing scaf.exe

From <https://stackoverflow.com/questions/50889647/best-way-to-get-exe-folder-path>

8.2.3 Variable Documentation

8.2.3.1 dirKey

```
const string dirKey = "templateDir"
```

The JSON key for the template directory. Should key a string.

8.2.3.2 infoKey

```
const string infoKey = "infos"
```

The JSON key for the saved infos. Should key an object where the keys for all values are strings.

8.3 Config.h

[Go to the documentation of this file.](#)

```
1 #pragma once
9 #include <string>
10 #include <filesystem>
11 #include <iostream>
12 #include <map>
13
14 using namespace std;
15 namespace fs = std::filesystem;
16
18 const string dirKey = "templateDir";
20 const string infoKey = "infos";
21
31 fs::path GetFullExePath();
32
42 class Config {
43     private:
45         fs::path configPath;
47         fs::path templateDir;
49         map<string, string> infos;
50
51     public:
53         Config();
57         Config(fs::path customPath);
58
60         fs::path getPath();
61
66         void readConfig();
67
69         fs::path getTemplateDir();
70
77         bool setTemplateDir(fs::path newdir);
78
80         bool hasTemplateDir();
81
83         string getInfo(string template_name);
84
89         void setInfo(string key, string value);
90
91
95         void writeConfig();
96 };
97
```

8.4 src/Filer.cpp File Reference

Definitions for [Filer](#) namespace.

```
#include "Filer.h"
#include <iostream>
#include <bits/stdc++.h>
```

8.4.1 Detailed Description

Definitions for [Filer](#) namespace.

Program entry point.

Author

Karl Miller

Date

April 2023

8.5 src/Filer.h File Reference

Declarations for [Filer](#) namespace.

```
#include <filesystem>
#include <map>
#include <vector>
#include <functional>
```

Classes

- struct [Filer::copy_result](#)
Provides data about a recursive copy operation.

Namespaces

- namespace [Filer](#)
Performs some filesystem operations and queries.

Functions

- copy_result [Filer::copyRecursive](#) (fs::path from, fs::path to)
- bool [Filer::isEmpty](#) (fs::path checkdir)
- bool [Filer::clearDir](#) (fs::path dirToClear)
- bool [Filer::fillMapWithDirectories](#) (const fs::path &p, map< string, fs::path > &m)
- bool [Filer::fillVectorWithDirectories](#) (const fs::path &p, vector< string > &m)
- bool [Filer::fillVectorWithDirectories](#) (const fs::path &p, vector< string > &v, string &filter_prefix)

8.5.1 Detailed Description

Declarations for [Filer](#) namespace.

Author

Karl Miller

Date

April 2023

8.6 Filer.h

[Go to the documentation of this file.](#)

```

1 #pragma once
2 #include <filesystem>
3 #include <map>
4 #include <vector>
5 #include <functional>
6 using namespace std;
7 namespace fs = std::filesystem;
8
9
10 namespace Filer {
11     typedef struct {
12         int filescopied;
13         int folderscopied;
14         bool gitskipped;
15     } copy_result;
16     copy_result copyRecursive(fs::path from, fs::path to);
17
18     bool isEmpty(fs::path checkdir);
19
20     bool clearDir(fs::path dirToClear);
21
22     bool fillMapWithDirectories(const fs::path &p, map<string, fs::path>& m);
23
24     bool fillVectorWithDirectories(const fs::path &p, vector<string>& m);
25
26     bool fillVectorWithDirectories(const fs::path &p, vector<string>& v, string &filter_prefix);
27 }

```

8.7 src/Scaf.h File Reference

Declarations for [Scaf](#) class.

```

#include <vector>
#include <filesystem>
#include "Config.h"
#include "Filer.h"

```

Classes

- class [Scaf](#)
Parses and executes the command line arguments.

Functions

- void [stringLower](#) (string &s)
- bool [promptYN](#) (bool default_yn)
- void [printCopyResult](#) ([Filer::copy_result](#) &copied)

8.7.1 Detailed Description

Declarations for [Scaf](#) class.

Author

Karl Miller

Date

April 2023

8.7.2 Function Documentation

8.7.2.1 printCopyResult()

```
void printCopyResult (
    Filer::copy_result & copied )
```

Prints a copy result on 3 lines with correct pluralization.

Parameters

<i>copied</i>	The result of a recursive directory copy.
---------------	---

8.7.2.2 promptYN()

```
bool promptYN (
    bool default_yn )
```

Looks for a 'y' or 'n'.

8.7.2.3 stringLower()

```
void stringLower (
    string & s )
```

Converts a string to all lower-case in place.

8.8 Scaf.h

[Go to the documentation of this file.](#)

```
1 #pragma once
2 #include <vector>
3 #include <filesystem>
4
5 #include "Config.h"
6 #include "Filer.h"
7
8 using namespace std;
9
10 void stringLower(string& s);
11
12 bool promptYN(bool default_yn);
13
14 void printCopyResult(Filer::copy_result & copied);
15
16 class Scaf {
17     public:
18         Scaf();
19         Scaf(filesystem::path config_path);
20         bool Start(int argc, char ** argv);
21 }
```

```
47
48     private:
49         Config config;
54         bool Help(int index, vector<string>& args);
55
60         bool Root(int index, vector<string>& args);
61
66         bool Add(int index, vector<string>& args);
67
72         bool Load(int index, vector<string>& args);
73
78         bool List(int index, vector<string>& args);
79
84         bool Info(int index, vector<string>& args);
85
90         bool Set(int index, vector<string>& args);
91
96         bool Remove(int index, vector<string>& args);
97
102         bool Rename(int index, vector<string>& args);
103
105         void printHelp();
106
108         void printHelpRoot();
110         void printHelpAdd();
112         void printHelpInfo();
114         void printHelpSet();
116         void printHelpRemove();
118         void printHelpLoad();
120         void printHelpRename();
122         void printHelpList();
123
124 };
```

8.9 src/Template.cpp File Reference

Definitions for [Template](#) class (deprecated).

```
#include "Template.h"
#include <filesystem>
```

8.9.1 Detailed Description

Definitions for [Template](#) class (deprecated).

Author

Karl Miller

Date

April 2023

Deprecated

Note

[Template](#) is not currently used in the program!

It may be utilized on a future iteration and refactor.

8.10 src/Template.h File Reference

Declarations for [Template](#) class.

```
#include <string>
#include <vector>
```

Classes

- class [Template](#)

Holds information about a template. Not used!

8.10.1 Detailed Description

Declarations for [Template](#) class.

Author

Karl Miller

Date

April 2023

Deprecated

Note

[Template](#) is not currently used in the program!

8.11 Template.h

[Go to the documentation of this file.](#)

```
1 #pragma once
10 #include <string>
11 #include <vector>
12
13 using namespace std;
14
27 class Template {
28     private:
30         string path;
32         string alias;
34         string info;
35
36     public:
37
38         Template(string a_path, string an_alias, string some_info);
39         Template(string a_path, string an_alias);
40         Template(string a_path);
41         Template();
42
44         string getPath();
45
47         string getAlias();
48
50         void setAlias(string an_alias);
51
53         string getInfo();
54
56         void setInfo(string some_info);
57
58 };
```

8.12 tests/ConfigTest.cpp File Reference

Provides tests for [Config](#). See [Config.h](#).

```
#include "QUnit.hpp"
#include <iostream>
#include <filesystem>
#include <fstream>
#include "Config.h"
```

Classes

- class [ConfigTest](#)
Tests the [Config](#) class.

Variables

- const string **tmp_conf** = "./tmp/testconf.json"
- const string **tmp_tdir** = "./tmp/tmp1"

8.12.1 Detailed Description

Provides tests for [Config](#). See [Config.h](#).

Author

Karl Miller

Date

April 2023

8.13 tests/FilerTest.cpp File Reference

Provides tests for [Filer](#). See [Filer.h](#).

```
#include <iostream>
#include <filesystem>
#include <fstream>
#include "QUnit.hpp"
#include "Filer.h"
```

Classes

- class [FilerTest](#)
Tests the [Filer](#) namespace.

Variables

- const string `flr_tmp_tdir` = `"/tmp/tmpl2"`
- const string `flr_tmp_tdir2` = `"/tmp/tmpl2/tt"`
- const string `flr_tmp_fl` = `"/tmp/tmpl2/t.x"`
- const string `flr_tmp_fl2` = `"/tmp/tmpl2/tt/t2.x"`
- const string `flr_tmp_odir` = `"/tmp/tout"`

8.13.1 Detailed Description

Provides tests for [Filer](#). See [Filer.h](#).

Author

Karl Miller

Date

April 2023

8.14 tests/TemplateTest.cpp File Reference

Provides tests for [Template](#). See [Template.h](#).

```
#include "QUnit.hpp"
#include <iostream>
#include "Template.h"
```

Classes

- class [TemplateTest](#)
Tests the [Template](#) class.

8.14.1 Detailed Description

Provides tests for [Template](#). See [Template.h](#).

Author

Karl Miller

Date

April 2023

Note

[TemplateTest](#) is deprecated! [Template](#) is not currently used in the program!

Deprecated

8.15 tests/testAll.cpp File Reference

Calls appropriate unit tests by parsing command line arguments.

```
#include <string>
#include <iostream>
#include "TemplateTest.cpp"
#include "ConfigTest.cpp"
#include "FilerTest.cpp"
```

Functions

- `int main (int argc, char **argv)`
Parses command line arguments to run all or one test.

8.15.1 Detailed Description

Calls appropriate unit tests by parsing command line arguments.

Author

Karl Miller

Date

April 2023

8.15.2 Function Documentation

8.15.2.1 main()

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

Parses command line arguments to run all or one test.

Runs the tests.

Format for CLI args are as follows:

```
test { template | all } { noisy | normal | quiet | silent }
```

Defaults to normal verbosity and all tests.

Index

- clearDir
 - Filer, [11](#)
- Config, [15](#)
 - Config, [15](#), [16](#)
 - getInfo, [16](#)
 - getPath, [16](#)
 - getTemplateDir, [16](#)
 - hasTemplateDir, [16](#)
 - readConfig, [16](#)
 - setInfo, [17](#)
 - setTemplateDir, [17](#)
 - writeConfig, [17](#)
- Config.cpp
 - GetFullExePath, [23](#)
- Config.h
 - dirKey, [25](#)
 - GetFullExePath, [25](#)
 - infoKey, [25](#)
- ConfigTest, [18](#)
- copyRecursive
 - Filer, [12](#)
- dirKey
 - Config.h, [25](#)
- Filer, [11](#)
 - clearDir, [11](#)
 - copyRecursive, [12](#)
 - fillMapWithDirectories, [12](#)
 - fillVectorWithDirectories, [13](#)
 - isEmpty, [13](#)
- Filer::copy_result, [18](#)
 - filescopied, [18](#)
 - folderscopied, [19](#)
 - gitskipped, [19](#)
- FilerTest, [19](#)
- filescopied
 - Filer::copy_result, [18](#)
- fillMapWithDirectories
 - Filer, [12](#)
- fillVectorWithDirectories
 - Filer, [13](#)
- folderscopied
 - Filer::copy_result, [19](#)
- getAlias
 - Template, [21](#)
- GetFullExePath
 - Config.cpp, [23](#)
 - Config.h, [25](#)
- getInfo
 - Config, [16](#)
 - Template, [22](#)
- getPath
 - Config, [16](#)
 - Template, [22](#)
- getTemplateDir
 - Config, [16](#)
- gitskipped
 - Filer::copy_result, [19](#)
- hasTemplateDir
 - Config, [16](#)
- infoKey
 - Config.h, [25](#)
- isEmpty
 - Filer, [13](#)
- main
 - testAll.cpp, [34](#)
- printCopyResult
 - Scaf.h, [29](#)
- promptYN
 - Scaf.h, [29](#)
- readConfig
 - Config, [16](#)
- Scaf, [19](#)
 - Scaf, [20](#)
 - Start, [20](#)
- Scaf.h
 - printCopyResult, [29](#)
 - promptYN, [29](#)
 - stringLower, [29](#)
- setAlias
 - Template, [22](#)
- setInfo
 - Config, [17](#)
 - Template, [22](#)
- setTemplateDir
 - Config, [17](#)
- src/Config.cpp, [23](#)
- src/Config.h, [24](#), [26](#)
- src/Filer.cpp, [26](#)
- src/Filer.h, [27](#), [28](#)
- src/Scaf.h, [28](#), [29](#)
- src/Template.cpp, [30](#)
- src/Template.h, [31](#)

Start

 Scaf, [20](#)

stringLower

 Scaf.h, [29](#)

Template, [21](#)

 getAlias, [21](#)

 getInfo, [22](#)

 getPath, [22](#)

 setAlias, [22](#)

 setInfo, [22](#)

TemplateTest, [22](#)

testAll.cpp

 main, [34](#)

tests/ConfigTest.cpp, [32](#)

tests/FilerTest.cpp, [32](#)

tests/TemplateTest.cpp, [33](#)

tests/testAll.cpp, [34](#)

writeConfig

 Config, [17](#)