Logger

Generated by Doxygen 1.10.0

1 Logger	1
1.1 Basic Logger	1
1.1.1 How to Use	1
1.1.2 How to log	2
1.1.3 Ease if use	2
1.1.4 Log Channels	2
1.1.5 Log Orders	2
1.1.6 File Output	2
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Class Documentation	7
4.1 Logger Class Reference	7
4.1.1 Member Enumeration Documentation	7
4.1.1.1 LogChannel	7
4.1.1.2 LogOrder	8
4.1.2 Member Function Documentation	8
4.1.2.1 Log() [1/2]	8
4.1.2.2 Log() [2/2]	8
4.1.2.3 setFilepath()	9
5 File Documentation	11
5.1 C:/Users/charl/OneDrive/Documents/OpenSource Attempts/DHLo/Answer/Logger.h File Reference .	11
5.2 Logger.h	11
5.3 C:/Users/charl/OneDrive/Documents/OpenSource Attempts/DHLo/Answer/README.md File Reference	13
Index	15

Logger

• ME

1.1 Basic Logger.

Task Description:

Create a universal trace library for application event logging to different channels. The library should support logging to at least one of the following channels: TCP, serial, console, file, etc. Additionally, it should be designed in a non-blocking manner to ensure it does not impede the normal operation of the applications utilizing it.

Requirements:

- 1. Implement a C++ library that enables application event logging.
- 2. The library should support logging to at least one of the following channels: TCP, serial, console, file, etc. You may choose the channel that best suits your expertise or demonstrate versatility by implementing logging to multiple channels.
- 3. Ensure the library is non-blocking to prevent it from interfering with the normal execution of applications.
- 4. Provide clear documentation on how to use the library and its different features.
- 5. Optionally, you may include unit tests to ensure the reliability and robustness of your implementation.

1.1.1 How to Use

This is a header-only library. Copy Logger.h file and #include it in your project.

2 Logger

1.1.2 How to log

```
Here is an example
#include "Logger.h"

int main()
{
    std::string a = "string";
    std::string str = "Test Log with std::string";
    std::string result;

    Logger::Log (Logger::ConsoleChannel , Logger::LogDebug, "Test Log with number %d" , 5);
    Logger::Log (Logger::ConsoleChannel , Logger::LogInfo, "Test Log with c-string %s , %i" , a.c_str(), 15);
    Logger::Log (Logger::ConsoleChannel , Logger::LogError, str);

    Logger::setFilepath("output.txt");
    Logger::Log (Logger::FileChannel , Logger::LogDebug, str);
    Logger::Log (Logger::FileChannel , Logger::LogInfo, "Test Log with c-string %s , %i" , a.c_str(), 15);
}
```

Output:

03-05-2024: 11:46:30 [Debug] Test Log with number 5 03-05-2024: 11:46:30 [Info] Test Log with c-string string, 15 03-05-2024: 11:46:30 [Error] Test Log with std::string

and file output

03-05-2024 : 11:46:30 [Debug] Test Log with std::string 03-05-2024 : 11:46:30 [Info] Test Log with c-string string , 15

1.1.3 Ease if use

No initialization, just use log functions

1.1.4 Log Channels

Theses are the channels currently supported

Logger::FileChannel
Logger::ConsoleChannel

1.1.5 Log Orders

Theses are the orders currently supported

Logger::LogDebug
Logger::LogInfo
Logger::LogError

1.1.6 File Output

To set the file path file output, call Logger::setFilepath("output.txt");

if no file path is set the default file location is "log.txt".

Class Index

2.1 Class List

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

4 Class Index

File Index

^ 4		
~~~		List
J. I	1 110	LISI

Here is a list of all files with brief descriptions:	
C:/   sers/charl/OneDrive/Documents/OnenSource Attempts/DHI o/Answer/Logger h	1:

6 File Index

### **Class Documentation**

#### 4.1 Logger Class Reference

```
#include <Logger.h>
```

#### **Public Types**

- enum LogChannel { FileChannel , ConsoleChannel }
- enum LogOrder { LogDebug , LogInfo , LogError }

#### **Static Public Member Functions**

- static bool setFilepath (const char *new filepath)
- static void Log (LogChannel channel, LogOrder order, std::string logMessage)
- template<typename... Args> static void Log (LogChannel channel, LogOrder order, const char *logMessage, Args... args)

#### 4.1.1 Member Enumeration Documentation

#### 4.1.1.1 LogChannel

enum Logger::LogChannel

The different channels currently supported.

#### Enumerator

FileChannel	
ConsoleChannel	

8 Class Documentation

#### 4.1.1.2 LogOrder

```
enum Logger::LogOrder
```

The different orders currently supported.

#### **Enumerator**

LogDebug	
LogInfo	
LogError	

#### 4.1.2 Member Function Documentation

#### 4.1.2.1 Log() [1/2]

Writes the log messge for the specifiesd order to the specified channel .

#### **Parameters**

channel	the channel to wwrite the log to, file or console
Order	the log order, debug, info, error
logMessage	the message to be logged in c-string format
args	the arguments that can be passed into the c-string

#### 4.1.2.2 Log() [2/2]

Writes the log messge for the specifies order to the specified channel .

#### **Parameters**

channel	the channel to wwrite the log to, file or console
Order	the log order, debug, info, error
logMessage	the message to be logged in string format

#### 4.1.2.3 setFilepath()

Set the file path and name to which the log output would be written.

#### **Parameters**

new_filepath	the filepath
micros	the microseconds fraction

#### Returns

true if the file exists or was created otherwise it returns false

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

 $\bullet \ \ C:/Users/charl/One Drive/Documents/Open Source\ Attempts/DHLo/Answer/Logger.h$ 

10 Class Documentation

### **File Documentation**

# 5.1 C:/Users/charl/OneDrive/Documents/OpenSource Attempts/DHLo/Answer/Logger.h File Reference

```
#include <string>
#include <cstdio>
#include <mutex>
#include <ctime>
```

#### Classes

class Logger

#### 5.2 Logger.h

#### Go to the documentation of this file.

```
00001
00003 * @file Logger.h
00007
00008
00009 #pragma once
00010 #include <string>
00011 #include <cstdio>
00012 #include <mutex>
00013 #include <ctime>
00014
00015
00016 //needed to use fopen if using visual studio
00017 #if defined(_MSC_VER)
00018 #define _CRT_SECURE_NO_WARNINGS 00019 #endif
00020
00021
00022 class Logger{
00023 public:
      Logonannel {
   FileChannel , ConsoleChannel
};
    enum LogChannel {
00028
00029
00030
00031
00036 enum LogOrder {
        LogDebug, LogInfo , LogError
```

12 File Documentation

```
00038
          };
00039
00040
00049
          static bool setFilepath(const char* new_filepath) {
00050
            return getInstance().init(new_filepath);
00051
00052
00061
          static void Log(LogChannel channel, LogOrder order, std::string logMessage) {
            getInstance().LOG(channel, order, logMessage.c_str());
00062
00063
00064
00074
          template<typename... Args>
static void Log(LogChannel channel, LogOrder order, const char* logMessage, Args... args) {
00075
00076
           getInstance().LOG(channel, order, logMessage, args...);
00077
00078
00079 private:
00080
          std::mutex logMutex;
00081
00082
          char _time[80];
          const char* timestampFormat = "%d-%m-%Y : %T";
std::FILE* file = 0;
00083
00084
          const char* filepath = "log.txt";
00085
00086
00087
          Logger() {}
00088
00089
          Logger(const Logger&) = delete;
00090
          Logger& operator= (const Logger&) = delete;
00091
00092
          ~Logger()
00093
          {
00094
              deleteFile();
00095
00096
00097
          static Logger& getInstance()
00098
00099
              static Logger instance;
00101
              return instance;
00102
          }
00103
00104
          const char* orderString(LogOrder order) {
             if (order == LogDebug) {
    return "[Debug] ";
00105
00106
00107
00108
              else if (order == LogInfo) {
00109
                 return "[Info] ";
00110
00111
              else if (order == LogError) {
                 return "[Error]
00112
00113
00114
              else {
                  return "[****]
00115
00116
              }
00117
         }
00118
00124
          bool init(){
             deleteFile();
00125
00126
              file = std::fopen(filepath, "a");
00127
              if (file == 0)
00128
00129
              {
00130
                  return false;
00131
00132
              return true;
00133
         }
00134
          bool init(const char* new_filepath) {
00142
00143
             deleteFile();
              file = std::fopen(new_filepath, "a");
00144
00145
              if (file == 0)
00146
00147
00148
                  return false:
00149
00150
              return true;
00151
         }
00152
00153
          void deleteFile()
00154
00155
              if (file)
00156
              {
00157
                   std::fclose(file);
00158
                  file = 0;
00159
          }
00160
00161
```

```
00162
          template<typename... Args>
          void fileLog(const char* time, const char* messageOrderStr, const char* logMessage, Args... args)
00164
              if (file)
            {
00165
                                           ", time);
                  std::fprintf(file, "%s
00166
                  std::fprintf(file, messageOrderStr);
00167
                 std::fprintf(file, logMessage, args...);
std::fprintf(file, "\n");
00168
00169
00170
00171
00172
             else {
                 if (getInstance().init())
00173
                      getInstance().fileLog(time, messageOrderStr, logMessage, args...);
00174
00175
        }
00176
00177
         template<typename... Args>
00178
         void consoleLog(const char* time, const char* messageOrderStr, const char* logMessage, Args...
     args) {
                                ", time);
00179
             std::printf("%s
00180
             std::printf(messageOrderStr);
00181
             std::printf(logMessage, args...);
             std::printf("\n");
00182
00183
         }
00184
00195
        template<typename... Args>
        void LOG(LogChannel channel, LogOrder order, const char* message, Args... args) {
00196
          std::time_t current_time = std::time(0);
00197
00198
             std::tm* timestamp = std::localtime(&current_time);
00199
           std::scoped_lock lock(logMutex);
std::strftime(_time, 80, timestampFormat, timestamp);
00200
00201
00202
00203
            if (channel == FileChannel) {
00204
                  getInstance().fileLog(_time, orderString(order), message, args...);
00205
             else if (channel == ConsoleChannel) {
00206
                 getInstance().consoleLog(_time, orderString(order), message, args...);
00208
00209
00210 };
00211
00212 //https://charlescookey.com/
```

# 5.3 C:/Users/charl/OneDrive/Documents/OpenSource Attempts/DHLo/Answer/README.md File Reference

14 File Documentation

### Index

```
C:/Users/charl/OneDrive/Documents/OpenSource
                                                At-
         tempts/DHLo/Answer/Logger.h, 11
C:/Users/charl/OneDrive/Documents/OpenSource
                                                At-
         tempts/DHLo/Answer/README.md, 13
ConsoleChannel
    Logger, 7
FileChannel
    Logger, 7
Log
    Logger, 8
LogChannel
    Logger, 7
LogDebug
    Logger, 8
LogError
    Logger, 8
Logger, 1, 7
    ConsoleChannel, 7
    FileChannel, 7
    Log, 8
    LogChannel, 7
    LogDebug, 8
    LogError, 8
    LogInfo, 8
    LogOrder, 7
    setFilepath, 8
LogInfo
    Logger, 8
LogOrder
    Logger, 7
setFilepath
    Logger, 8
```