Logger library (Connected Robotics test)

Short Description

This library is the version 0.1 of a multipurpose logging library, which includes several log levels (Info, Debug, Warning, Error and Fatal) and allows incluiding variables to the log messages. This library has been tested on Ubuntu 18.04 with C++11.

Installation

In order to install follow the following steps:

1. Uncompress the zip file 2. Navigate to the project folder *{initial folder}/logger_cpp/logger* ```bash cd {initial folder}/logger_cpp/logger ``` 3. Create a *build* folder and go to it. ```bash mkdir build cd build ``` 4. Compile and install the library ```bash cmake .. sudo make install ``` You should expect to see the following output: ```bash $: ~/logger_cpp/logger/build$ sudo make install

```
Scanning dependencies of target Logger
[ 20%] Building CXX object CMakeFiles/Logger.dir/src/logger.cc.o
[ 40%] Linking CXX shared library libLogger.so
[ 40%] Built target Logger
Scanning dependencies of target Tests
[ 60%] Building CXX object CMakeFiles/Tests.dir/tests/test1.cc.o
[ 80%] Building CXX object CMakeFiles/Tests.dir/src/logger.cc.o
[100%] Linking CXX executable Tests
[100%] Built target Tests
Install the project...
-- Install configuration: ""
-- Installing: /usr/local/lib/libLogger.so.0.1
-- Up-to-date: /usr/local/lib/libLogger.so.1
-- Up-to-date: /usr/local/lib/libLogger.so
-- Installing: /usr/local/include/logger.h

```

```

5. You can now perform automated tests to confirm that the library works properly. Here, the library will be tested on multiple threads, for different types of levels and messages.

   ```
   ./Tests
   ```

   You should get the following result:
   ```

```
===============================================================================
All tests passed (209 assertions in 8 test cases)
```

You can also see a .log file generated by the tests in ./log/default.log

6. Now you can use the library (please keep in mind that you have to link the .so file). You probably will need to run:

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib
```

Using the library

Before using the library, please create a config file like the following:

```
#Absolute path where logs will be saved (Comment for default folder)
LogFilePath: /home/jdgalviss/logger_cpp/logger/log/file.log
#Enable Logging to Terminal?
LogToTerminal: true
#Enable Logging to File?
LogToFile: true
```

Now in your code you have to define the count_id on which you want the logst to start. For example:

```
int cr::Logger::log_count_ = 1;
```

You also have to create a Logger object and pass it the path to your .config file:

```
cr::Logger * logger = new cr::Logger(config_path);
```

Now you can log files:

```
logger->Info("This is an %v example", variable);
logger->Warning("This is a Warning example");
logger->Debug("This is the number %v Debug example", 1);
logger->Error("This is an Error example");
```

Which should produce:

```
1 [INFO]: Mon Feb 17 03:02:53 2020:This is an Info example
2 [WARNING]: Mon Feb 17 03:02:53 2020:This is a Warning example
```

```
3 [DEBUG]: Mon Feb 17 03:02:53 2020:This is the number 1 Debug example
4 [ERROR]: Mon Feb 17 03:02:53 2020:This is an Error example
```

Samples

You can compile the samples by following the next steps:

1. Go to the samples folder

```
cd ./examples
```

2. Compile the samples

```
mkdir build
cd build
cmake ..
cmake --build .
```

3. This should generate 2 executables (*Sample1* and *logging_threads*) which you can try:

```
$ ./Sample1
[Logger Configuration]: Trying to open configuration...
[Logger Configuration]: Unable to open configuration file [/usr/local/include/../conf
[Logger Configuration]: logging to file: false
[Logger Configuration]: logging to terminal: true
[Logger Configuration]: log directory: /home/jdgalviss/logger_cpp/logger/include/../l
[Logger Configuration]: logging to terminal: true
[Logger Configuration]: logging to file: false
[Logger Configuration]: Configuration complete

1 [INFO]: Mon Feb 17 03:02:53 2020:This is an Info example
2 [WARNING]: Mon Feb 17 03:02:53 2020:This is a Warning example
3 [DEBUG]: Mon Feb 17 03:02:53 2020:This is the number 1 Debug example
4 [ERROR]: Mon Feb 17 03:02:53 2020:This is an Error example
```

A simple program that implements this library looks as follows:

```
#include "logger.h"

int cr::Logger::log_count_ = 1;     //Initialize the log count

int main(){
    cr::Logger * logger = new cr::Logger(); //Logger object
    std::string variable = "Info";
```

```
logger->Info("This is an %v example", variable);
logger->Warning("This is a Warning example");
logger->Debug("This is the number %v Debug example", 1);
logger->Error("This is an Error example");
```