Facebow File Reader

Solution Overview

Introduction

This document is to outline the following technical assessment implementation:

Task 1: *Write a set of unit tests that test the functionality and correctness of the FacebowFileReader - what and how many tests is left up to your judgement.*

Task 2: *Write a CI script that automatically runs these tests on a CI system/portal of your choice.*

IDE: Visual Studio Community Version: 17.9.5 (Latest)

Issues

Issue #1 (LNK1104 gdi32.lib)

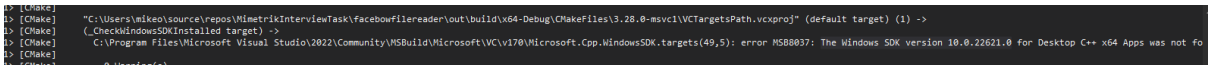
Fix:

- Adjust the CMake settings, set the CMake generator to 'Visual Studio 17 2022 Win64'
- Using x64 Developer Tools Command Prompt I ran 'vcpkg integrate install'

Notes:

- Linking in CMakeLists.txt did not work.
- Including the path to 'Gdi32.lib' in the system PATH variable did not work.

Issue #2 (Wrong Windows SDK Version: 10.0.22621.0)



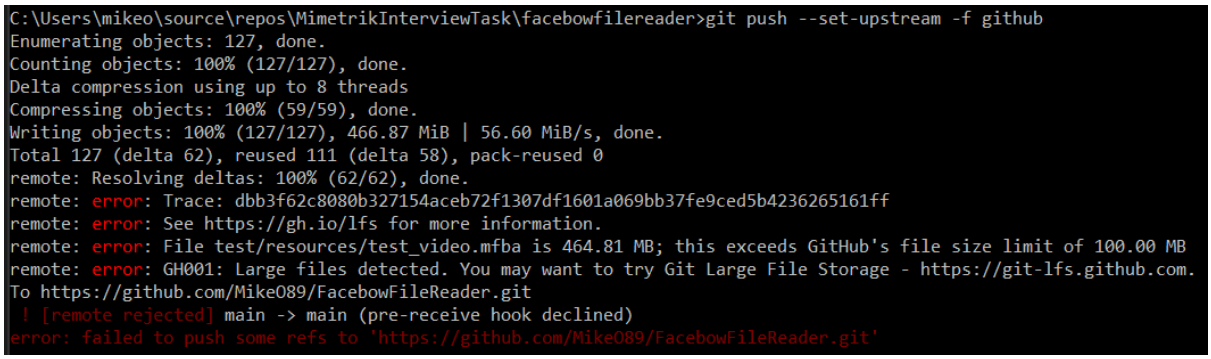
Fix:

- Removed any trace of the incorrect Windows SDK version on the system: success.

Notes:

- Attempted to specify the version in the CMakeSettings JSON, to no avail.

Issue #3 (Testing file too large for Github)



Fix:

- Stripped the ‘MFBA’ file down to 16 frames, bringing the file size under the 100MB limit.

Notes:

- Using Git’s Large File Storage is not an option as a symbolic link will be created in place of the file within the repository and when the workflow is running, the tests will open the symbol link files instead of the actual files pointed at.

Unit Tests

Library: GTest

Command: ctest --output-on-failure

- #1 Ensure an error is thrown on an empty ‘.mfba’ file.
- #2 Ensure an error is thrown when the file does not exist.
- #3 Error is thrown when the MFBA file signature is incorrect.
- #4 Error is thrown when the MFBA file version is set to ‘2.0.0’ .
- #5 An error is not thrown when the file headers are valid.
- #6 Errors are thrown when requesting data out-of-range.
- #7 File bytes are read correctly.
- #8 Ensure frame meta data is correctly loaded.
- #9 Ensure frame data is correctly loaded.
- #10 Ensure frame reading latency is sufficient.

Github Workflows

There are two workflows on Github: one for Ubuntu, the other for Windows. The repository URL: https://github.com/MikeO89/FacebowFileReader

Thoughts

- Start with tests for basic handling before expanding to more complicated testing.

- Any testing framework available with vcpkg will do due to the use being a one-off and not something that will be used in production (GTest by Google will likely be a safe-bet anyway given the creator is Google).

- FacebowFileReader would be better compiled as an actual library with the function implementations moved to a CPP file for obscurity.

- The file format for FacebowFileReader could be improved by storing the encryption method in the header information, allowing for multiple encryption methods as a different type of encoding may be required in the future. In addition, a compression method could could be encompassed to reduce the latency on transport as well as reducing the storage size.

Review

What went well?

1. Researched CMake and learned a good amount about it.

2. Thoroughly enjoyed working with low-level data again.

3. Learned a new unit test framework (GTest).

4. Integrating GTest with CMake was straight-forward.

5. The instructions and source code were easy to understand and follow.

6. Gained familiarity with Github Actions, which I have not used for Continuous Integration before.

What went not-so-well? / Limitations

1. Some initial issues with the CMake set up – as described above – but they were resolved. Issue #2 was due to erroneously installing the Windows 11 SDK a long while back and CMake was defaulting to this version instead of the version for Windows 10 (10.0.20348.0).

2. Unable to mock member function calls for the FacebowFileReader class due to member functions not being virtual (without resorting to hooking) – unable to make virtual due to assignment instructions not indicating being able to adjust FacebowFileReader.

3. Perhaps the way I have achieved the modifications in CMake are not best practice due to having minimal exposure to CMake in the past.

4. Did not take the time to learn GTest beyond basic use, and did not take the time to establish if there is a better framework to utilise.

What would I do different?

1. Spend more time to research best practices for CMake.

2. Explore GTest further, as well as compare GTest to other testing frameworks to ensure the best framework is chosen.

3. Seek clarification on whether the FacebowFileReader class can be modified to allow for it to be mocked.