**HMI Unit Test 2019 - TATA PROJECT**

**TATA Unit Test Documentation.**

Unit Testing

Version 1.0

**Author**- Rajat Gupta Initial Version Documentation

**Disclaimer**- This Documentation is specifically for Tata Project which is based on Linux Platform with QT Framework.

**Introduction**.

Unit Test Implementation is introduced in TATA X445 Project using GTest(Google Test), GMock (Google Mock) and LCov. The former one’s helps to write Test Cases with Different pre-defined MACROS whereas the latter one Analysis the Test cases execution and generates the Coverage Report. This Unit Test Documentation is designed for step by step process to implement the test cases and generate the Coverage report.

**Abbreviations- Used.**

|  |  |
| --- | --- |
| HMI | Human Machine Interface |
| GTest | Google Test |
| GMock | Google Mock |
|  |  |
|  |  |
|  |  |

**About Gtest:**

GTest is a Testing Framework also known as Google Test which provides different functionalities to Test the code in C++ feasibly. It provides multiple TEST MACROS to use as per requirement and need.

Now Question arises, why GTest?

1. Flexible to use as per requirement.
2. Repeatable.
3. Provides different methods for Different Parameters including Corner cases and also those which can fail the real time execution.
4. Easy to understand.

**GMOCK**

In may scenarios, it happens when Developer/Tested cannot Test some real time cases like index out of bound, Memory leak and Memory Overflow. Here comes the GMock to help you to test those cases. It seamlessly works with GTest to provide full Coverage for the code.

**Why GMock?**

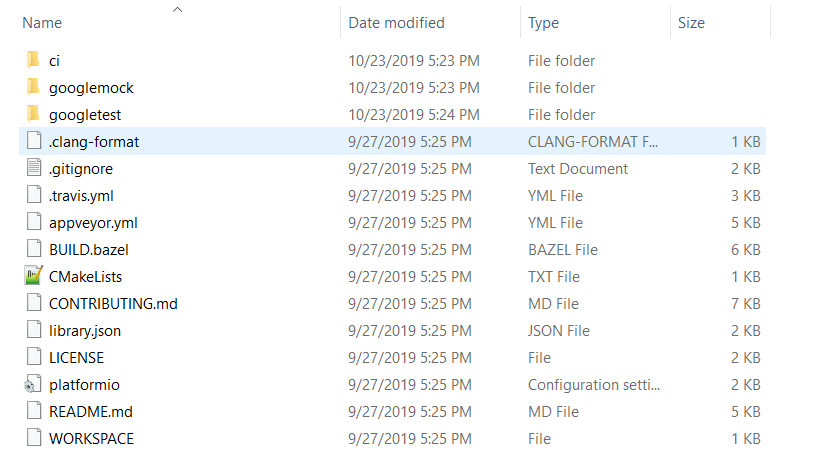
1. Supplies non-deterministic results (e.g. the current time)
2. Has states that are difficult to create or reproduce (e.g. a network error)
3. Is slow (e.g. a complete database, which would have to be initialized before the test)
4. Database overflow/ underflow.

**Starting with GTest and GMock Libraries.**

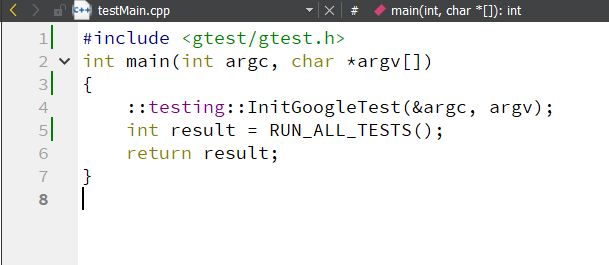
First, download the Gtest source code from google (open source):-

<https://github.com/google/googletest>

Below image shows the files/folder included in GTest downloaded folder.



The testMain.cpp should look like this:-

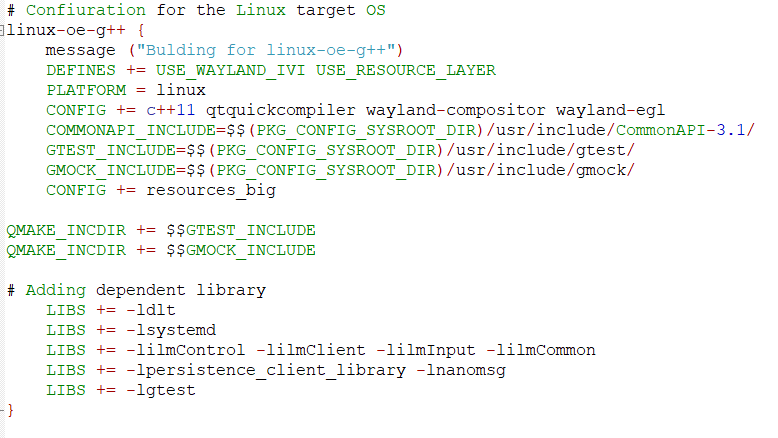


Modification as Per Windows/Linux Platform.

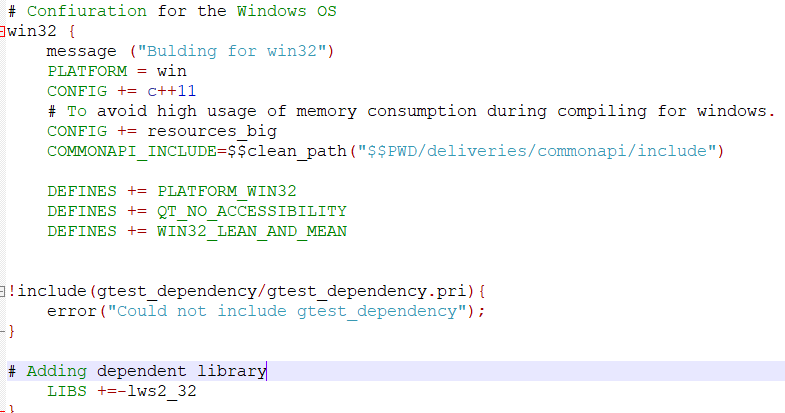
We need to include libraries as per the running platform. This should be included in .pro file of QT for the easy running on any platform with proper conditions/check.

**Changes in .Pro file**

* **For Linux :-**



**For Windows :-**



Also add

QMAKE\_CXXFLAGS += --coverage

QMAKE\_LFLAGS += --coverage in .pro file

**These flags enable extra information to be stored in the test binary that enables the Gcov mechanism.**

Whenever a new unit test file is added make sure that it is also added in the.pro file in the below tag:

SOURCES += main.cpp \

Write test cases in a separate file with the TEST MACROS using ASSERT, EXPECT etc.

Now for Windows, run the test cases by simply running the application and you will see the test cases getting executed with results i.e FAILED or PASSED. But report will be generate when build on server.

**Build Process.**

**Generating .gcno and .o files for coverage report.**

Now place your project on server having lcov and Gtest already installed.

Build the project by making .pro file.

It will generate gcno, object and moc files for the project at the same place where build files are located.

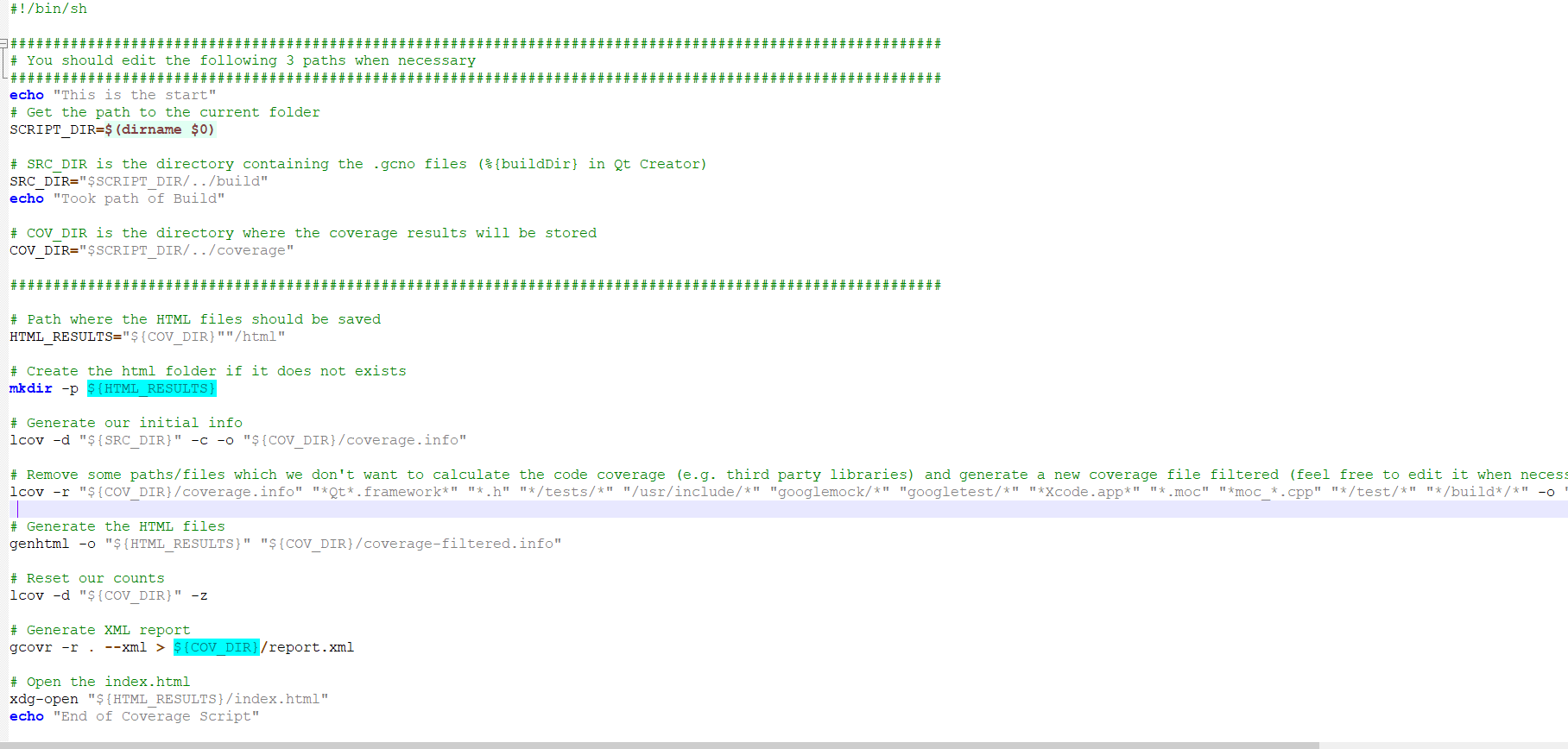
**Generating .gcda files.**

Now place the binary on target and also lib file needed to execute the binary. Restart the system and login to the Target. This will automatically generate the .gcda files needed to generate coverage report at the same directory for your build.

Ie. For ex /home/hmi/Rajat/build/ as it is the same location for my build files on the build server.

**Generating Coverage report.**

Now place all .gcda files into the build folder where .gcno, .o files are located after the build (build folder should be parallel placed with one directory back to the .pro file as can see the reference from **coverage.sh** script). Now run the coverage.sh file located in x451/ folder. This will generate the coverage report by combining all the .gcno, .gcda and .o files. Report will be in the **coverage** folder placed parallel to the directory of build folder. Snippet of coverage.sh file is given below.



Through the coverage report you can easily analyze the files, lines, functions covered through GTest.

**Things remaining**

* **Running on Live Target the test cases and seeing on Tera term. AS given in MMT help guide.**
* **Installation of LCOV and GTEST on build server and linkage of lib files**
* **Linux platform (for local machine)changes for the project and linkages also**