GoogleTest

SETUP W/ CMAKE

Imagine we have the following directory structure…

/

Src/ # source code

Tests/ # test suite

Vendor/ # google test location

CMakeLists.txt

/CmakeLists.txt

cmake\_minimum\_required(VERSION 3.17)

set(CMAKE\_CXX\_STANDARD 17)

set(PROJECT\_NAME myLib)

add\_subdirectory(${PROJECT\_NAME}/vendor/googletest)

# other setup here

add\_library(${PROJECT\_NAME} STATIC src/main.cpp) # library. Not executable!

/tests/CMakeLists.txt

cmake\_minimum\_required(VERSION 3.17)

set(PROJECT\_NAME tests)

add\_executable(${PROJECT\_NAME} test.cpp) # notice this is executable

target\_link\_libraries(${PROJECT\_NAME} PUBLIC gtest\_main) # gtest\_main important!!

As we have included gtest\_main, **we do not have to define our own main function.**

/tests/test.cpp

#include <gtest/gtest.h>

TEST(testSuiteName, testName) { // simple single test

EXPECT\_TRUE(true);

}

ASSERTIONS

ASSERT\_\* and EXPECT\_\* have the same suffixed names, the difference is, ASSERT\_\* generate fatal failures when they fail, and abort the current function. EXPECT\_\* versions generate non-fatal failures which don’t abort the current function.

|  |  |  |
| --- | --- | --- |
| EXPECT\_TRUE(val) | EXPECT\_EQ(val1, val2) | EXPECT\_EQ(val1, val2) |
| EXPECT\_NE(val1, val2) | EXPECT\_LT(val1, val2) | EXPECT\_GT(val1, val2) |
| EXPECT\_LE(val1, val2) | EXPECT\_GE(val1, val2) | EXPECT\_STREQ(str1, str2) |
| EXPECT\_STRNE(str1, str2) |  |  |

TEST FIXTURES

If you find yourself writing 2 or more tests that operate on similar data, you can use a test fixture; TEST\_F() instead of TEST(). This allows you to reuse the configurations of objects for several different tests.

Class FooTest : public ::testing::Test {

protected:

FooTest() {}

~FooTest() override {}

void SetUp() override {}

void TearDown() override {}

// class members declared here can be used by all tests in the test suite

}

TEST\_F(FooTest, doesXyx) {

// assertions

}

TEST\_F(FooTest, methodBarDoesAbc) {

Foo f;

EXPECT\_EQ(f.bar(), 0);

}