Use GTest for Unit Test

Install GTest

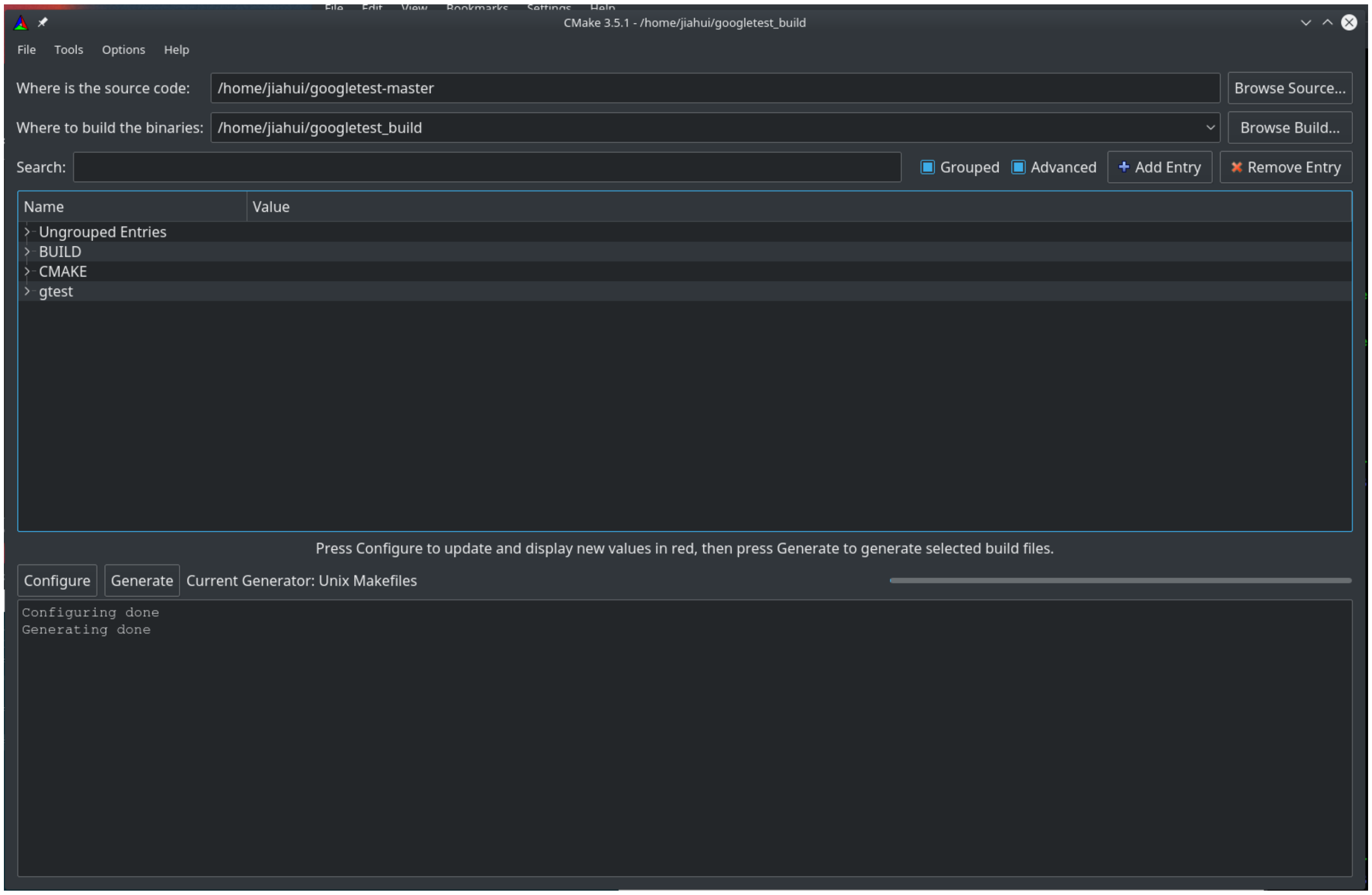
Download GTest from Git repo.

Download the zip file from https://github.com/google/googletest.git

Extract the source code from the zip file

unzip googletest-master.zip

Use CMake configure the software (Fig. 1)



Where is the source code: /home/jiahui/googletest-master

Where to build the binaries: /home/jiahui/googletest_build

Search:

Ungrouped Entries
BUILD
CMAKE
gtest

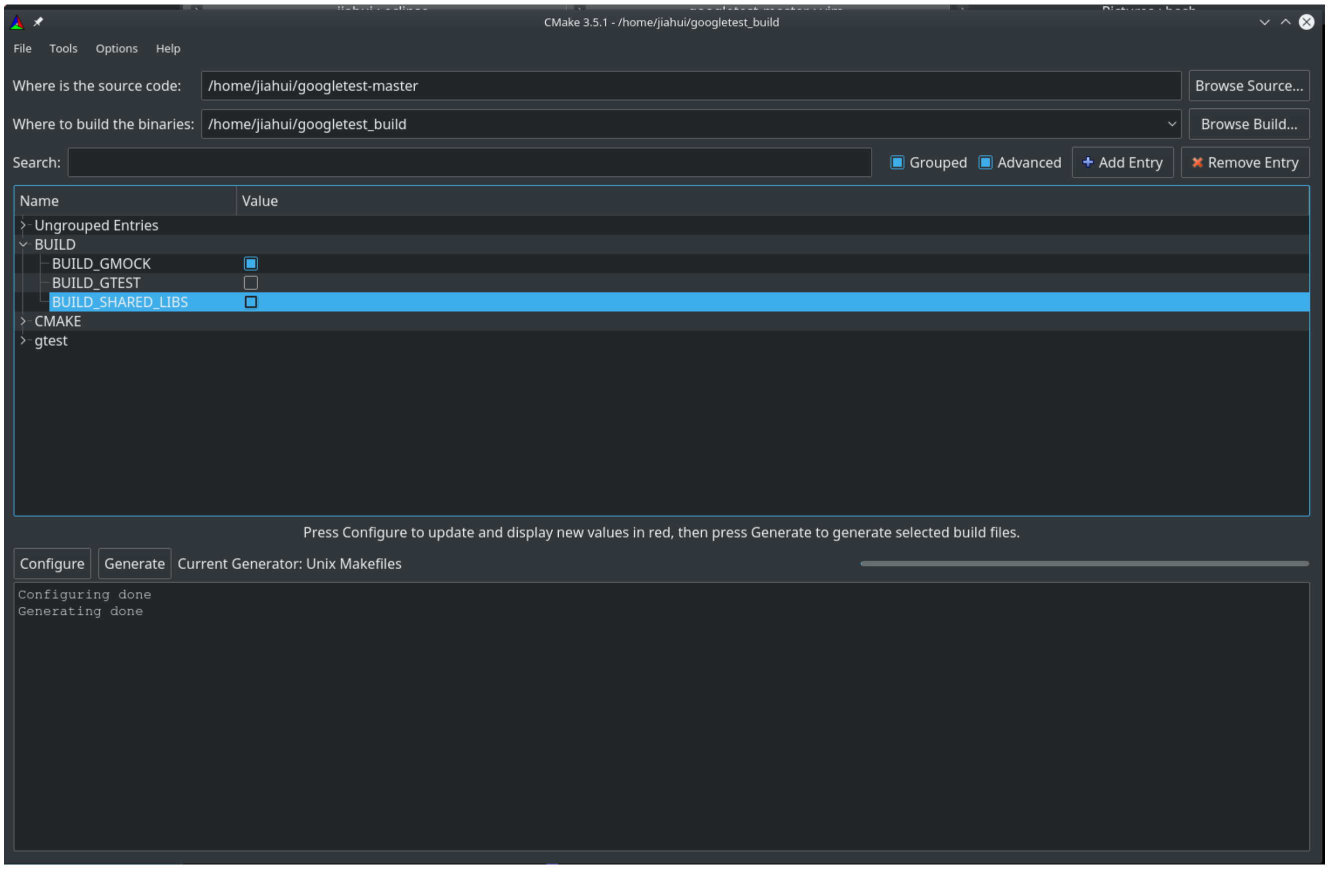
Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: Unix Makefiles

```
Configuring done
Generating done
```

Fig. 1 Use CMake to configure google test

- Build the library dynamically or statically by check / uncheck BUILD_SHARED_LIBS (Fig. 2)



Where is the source code: /home/jiahui/googletest-master

Where to build the binaries: /home/jiahui/googletest_build

Name	Value
Ungrouped Entries	
BUILD	
BUILD_GMOCK	☑
BUILD_GTEST	☐
BUILD_SHARED_LIBS	☐
CMAKE	
gtest	

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: Unix Makefiles

```
Configuring done
Generating done
```

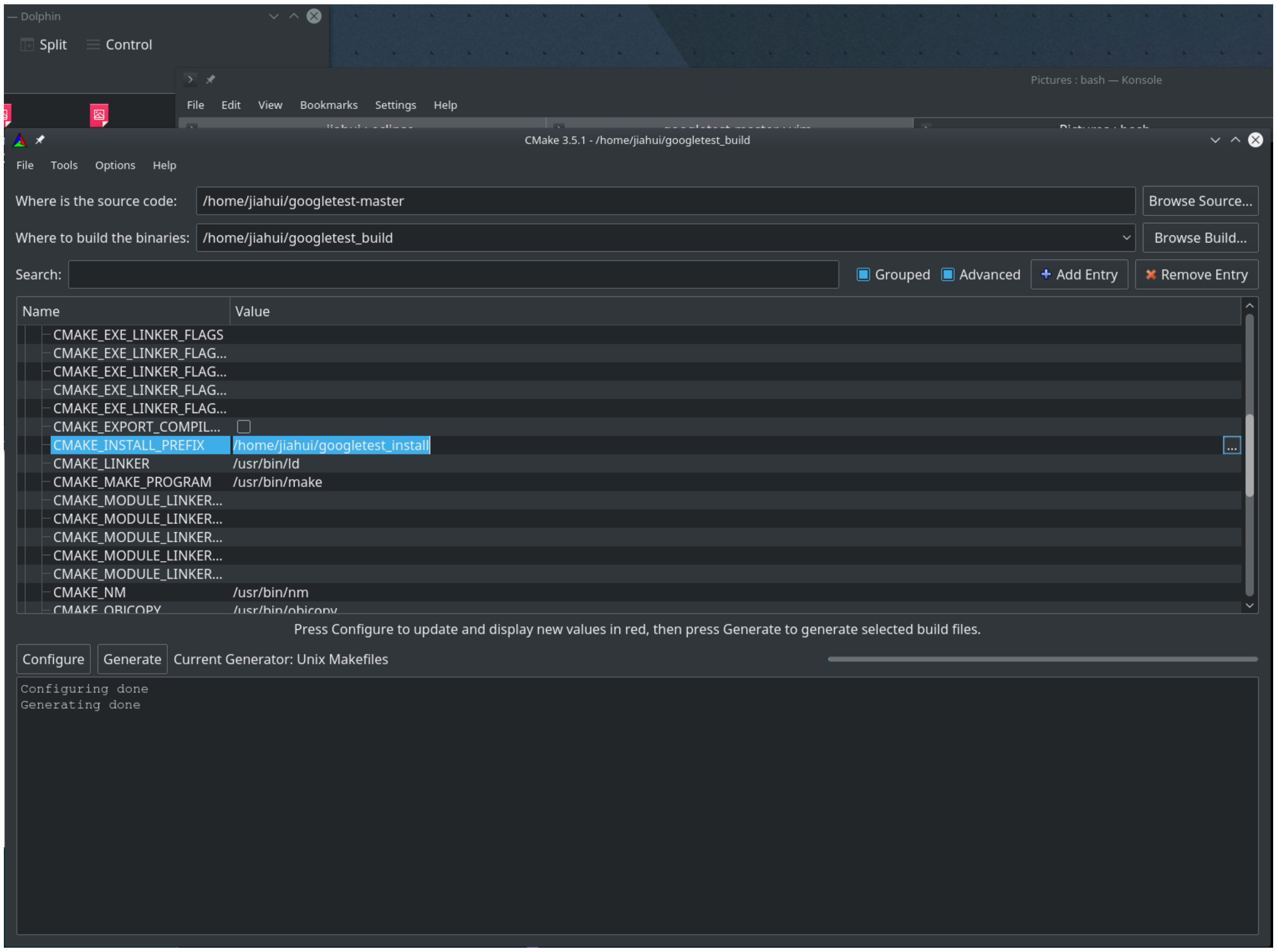
Fig. 2 Build google test as shared or static library

Click “Configure” and “Generate” .

And then run
make
make install

- Specify the installation path using CMAKE_INSTALL_PREFIX (Fig. 3)

generated by haroopad



CMake 3.5.1 - /home/jiahui/googletest_build

Where is the source code: /home/jiahui/googletest-master

Where to build the binaries: /home/jiahui/googletest_build

Search:

Name	Value
CMAKE_EXE_LINKER_FLAGS	
CMAKE_EXE_LINKER_FLAG...	
CMAKE_EXE_LINKER_FLAG...	
CMAKE_EXE_LINKER_FLAG...	
CMAKE_EXE_LINKER_FLAG...	
CMAKE_EXPORT_COMPIL...	☐
CMAKE_INSTALL_PREFIX	/home/jiahui/googletest_install
CMAKE_LINKER	/usr/bin/ld
CMAKE_MAKE_PROGRAM	/usr/bin/make
CMAKE_MODULE_LINKER...	
CMAKE_MODULE_LINKER...	
CMAKE_MODULE_LINKER...	
CMAKE_MODULE_LINKER...	
CMAKE_MODULE_LINKER...	
CMAKE_NM	/usr/bin/nm
CMAKE_OBJCOPY	/usr/bin/objcopy

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: Unix Makefiles

```
Configuring done
Generating done
```

Fig. 3 Specify the library installation path

Create test code

- Create code (e.g., AnalyzerLib_Unit_Testing.cpp) for the test of a c++ unit (project). The code is saved in the save of the source code of the unit (e.g., /Simmune/analyzerlib/AnalyzerLib_Unit_Testing.cpp). Below is the example of the test code

```
#include <string>
#include <gtest/gtest.h>
#include "Mongo/MongoInterface.h"

using ::testing::Test;
using ::testing::InitGoogleMock;
using ::testing::InitGoogleTest;

class MongoInterfaceTesting : public Test
```

```
{
    // Test fixture class with pre and post processing of testing
public: // methods
    void SetUp()
    {
        // Code will execute before the testing
        std::string host = "127.0.0.1";  // local connection
        int port = 10080;
        std::string user = "user";
        std::string pw = "password";
        std::string database = "Simmune";
        bool authAdmin = true;
        this->m_MongoInterface = new MongoInterface(host, port, user, pw, database, authAdmin)
    }
    void TearDown()
    {
        // Code will execute after the testing
        delete this->m_MongoInterface;
    }

public: // variables
    MongoInterface *m_MongoInterface;
};

TEST_F(MongoInterfaceTesting, Test_getDbName)
{
    // Testing of function getDbName() of MongoInterface class
    // getDbName() returns the name of database. Will test whether
    // it returns the same database name and whether the function
    // call throw out an exception
    std::string database = "Simmune";
    EXPECT_EQ(database, m_MongoInterface->getDbName());
    EXPECT_NO_THROW(m_MongoInterface->getDbName());
}

TEST_F(MongoInterfaceTesting, Test_setDbName)
{
    // Testing of function setDbName() of MongoInterface class
    // setDbName() assign the name of database (argument).
    // Will test whether getDbName() returns the expected database
    // name.
    std::string database = "myDatabase";
    m_MongoInterface->setDbName(database);
    EXPECT_EQ(database, m_MongoInterface->getDbName());
}

int main(int argc, char *argv[])
{
    // main function of the testing
    InitGoogleTest(&argc, argv);
    return RUN_ALL_TESTS();
}
```

- Add the following statements into the CMakeLists.txt of the unit.

```


INCLUDE_DIRECTORIES("/home/jiahui/work/mongodb_gtest_bin/googletest-src/googletest/include")

LINK_DIRECTORIES("/home/jiahui/work/mongodb_gtest_bin/googletest-build/googlemock/gtest") # add

ADD_EXECUTABLE(AnalyzerLib_Unit_Testing AnalyzerLib_Unit_Testing.cpp)
TARGET_LINK_LIBRARIES(AnalyzerLib_Unit_Testing  ${Boost_LIBRARIES} vtkCommonCore vtkFiltersCore

TARGET_LINK_LIBRARIES(AnalyzerLib_Unit_Testing pthread)
TARGET_LINK_LIBRARIES(AnalyzerLib_Unit_Testing gtest gtest_main)
```

Run test

An executable file (e.g., AnalyzerLib_Unit_Testing) will be created after build the unit (project). Run the executabnle file to implement the test.

generated by haroopad