Create Your First Legacy C/C++ Test Project with CppUTest

This paper describes how to integrate CppUTest based testing with your production code using the GCC toolchain environment.

1) Install and build CppUTest

2) Define CPPUTEST_HOME
Point it to the root directory of CppUTest. If you don't, the starter project makefile will not be able to find MakefileWorker.mk

3) Build the starter project
From a terminal window, change the directory to the root of the starter project. The same directory where this file was found. The make all.

```
cd /some/path/to/cpputest-starter-project
make all
```

You should see output like this:

```
compiling FooExampleFFFTests.cpp
compiling FooExample.fff.c
compiling io_CppUMock.cpp
compiling io_CppUMockTest.cpp
compiling fff_globals.c
compiling AllTests.cpp
compiling ExampleTest.cpp
compiling io.c
compiling Example.c
Building archive lib/libexample.a
a - obj/example-platform/io.o
a - obj/example-src/Example.o
Linking example_tests
Running example_tests
..............
OK (14 tests, 14 ran, 47 checks, 0 ignored, 0 filtered out, 0 ms)
```

4) Copy starter test files to your production code repository

Test files and production code files should all be kept in version control. Assuming your code is already in version control (if not, why not?!), this section describes how to integrate testing into your source repository.

Let's presume a directory structure like this:

```
ProductRepository
    I-- include
    I-- source
```

Copy cpputest-starter-project directories into your ProductRepository. It works best if the makefile does not have to use ".." to get to your source files. Using ".." for include files is no problem.

```
cd /some/path/to/cpputest-starter-project
./copy-to-product-repo.sh /some/other/path/to/ProductRepository
cd /some/other/path/to/ProductRepository
```

4) Build the initial tests in the product code repository

Open a terminal in the product code repository and make the example tests

```
cd /some/other/path/to/ProductRepository
make
```

You should get the same test build output as before.

Things that can go wrong:

- Your build fails because it cannot find CppUTest include or MakefileWorker.mk files.
  - Define the environment variable CPPUTEST_HOME to define the location of CppUTest.

5) Create an eclipse project - optional

1. To avoid issues with Eclipse, rename (or deleted – I deleted) existing .cproject, .project before performing the next steps
2. Open your eclipse workspace.
3. Create a new eclipse makefile project
    - Eclipse->File->New->Makefile Project with Existing Code

4. Enable automatic build on save
    - Eclipse->Project->select Build Automatically
    - Eclipse->Project->Properties->C/C++ Build->
    - Go to the “Behavior” tab and select check the box next to “Build on Resource Save”

How do you know you are done?

- When you see the makefile compilation and test output in the Console window.

Things that can go wrong:

- You already have Eclipse .project and .cproject files from your silicon vendors custom Eclipse implementation
    - get some help from your coaches -- sorry. For now just do command line test builds for now.

You are ready to add your first test. Now go look at the part two of the instructions. Keep working in small verifiable steps.