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

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

1) Install gcc toolchain
In Ubuntu

```
sudo apt-get install build-essential
```

In windows, I find cygwin (http://www.cygwin.com/) is the least trouble, The install may take a couple hours. Make sure to select the 'Devel' package in the installer.

2) Download, Install and build CppUTest
Download the latest from cpputest.org. It is best to put it into a directory near your production code so it can be checked into your source repository.

NOTE: My starter kit is not compatible with the install method described on cpputest.org. You cannot 'apt-get install cpputest' for use with my starter kit. Please install it as follows:

```
cd /close-to-your-production-code/tools/cpputest
./configure
make check
make tdd
```

You should see CppUTest's tests run. If you get errors, they are often easy to fix by looking at the error message. Often it is a matter of disabling some warning. You can also check with me or the cpputest google group. Please let me know there is a need for a change these directions.

3) Define CPPUTEST_HOME
Point CPPUTEST_HOME to the root directory of CppUTest. If you don't, the starter project makefile will not be able to find MakefileWorker.mk and the needed include files.

```
export CPPUTEST_HOME=/close-to-your-production-code/tools/cpputest
```

Under cygwin, you can use a windows environment variable.

4) Move the starter project
Move the starter project folder so that it is in the source repository with your production code. You want to be able to conveniently access your production code files and dependencies using relative paths. For example `/close-to-your-production-code/cpputest-starter-project`.

5) 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 /close-to-your-production-code/cpputest-starter-project
make all
```

You should see output announcing each file compiling and finally running the tests like this:

```
compiling blah.cpp
//one line for each file compiled
Building archive lib/libexample.a
//etc
Linking example_tests
Running example_tests
..............
OK (14 tests, 14 ran, 47 checks, 0 ignored, 0 filtered out, 0
ms)
```

6) You are ready to add your first test. Now go look at this page:

- https://wingman-sw.com/articles/get-your-legacy-c-into-a-test-harness

Keep working in small verifiable steps.