Create Your First Legacy C/C++ TEST

Now that you have the test environment in place, you are ready to get some of your production code under test. You should have completed and understood ReadmePart1_Gcc.rtf or ReadmePart1_VisualStudio.rtf before getting here. If you did, you are ready to start the Crash-to-Pass algorithm described in this article http://www.renaissancesoftware.net/blog/archives/27. If you have not done part1, you better go back and do it.

Get Your First Test to Compile

Choose a function (or class) that you want to test in CppUTest. Add a call to (or construct) it in MyFirstTest.cpp's TEST case. It won't compile. Add header files one at a time until you get the test case to compile. You will likely have to adjust the include path so the compiler can find the referenced include files.

Don't start with compiling the production code. The steps to get the test case to compile are usually a step in the right direction to getting the production code compiled.

How do you know you are done?

- When you finally see a linker error reporting the unresolved external reference to the code under test.

Potential problems:

- Target header has special keywords that do not compile using gcc (like cregister and interrupt)
  - This article shows how to make things like cregister and interrupt go away for off-target testing
  - http://www.renaissancesoftware.net/blog/archives/249
  - This article shows how to make 'asm' directives go away (also to test that they happen)
  - http://www.renaissancesoftware.net/blog/archives/136
- The header file cannot be compiled off-target (and you've eliminated the above approaches)
  - This article shows how to stub a header file.
  - http://www.renaissancesoftware.net/blog/archives/231
- Warnings are killing the compile.
  - Change the warning level in the Makefile.

Get a Production Code File to Compile Off-Target

Now that you have the test case compiled, it's time to get a single production code file into your test build. In visual studio, just add the existing item to the production code library project. For GCC, add the relative path to the production source file in the Makefile using the SRC_FILES variable.

What usually happens at this point, is that more directories have to be added to the include path. Hang in there, this can take a while

How do you know you are done?

- When the production code source file compiles without errors and now the linker cannot find some production code dependency.

Potential problems:

- See potential problems in the previous section

Get Your Production Code and Test to Link

To resolve linker errors you have to decide should the depended upon code be included in the build or faked out?

- Include the depended upon code
  - Pick a file and go back to 'Get a Production Code File to Compile Off-Target'
- Fake out the depended upon code. Make the simplest stub. Initially, put the test stub right into the test case. You can move it later and make it more useful. Don't worry initially about spies, mocks or fakes. Just stub it.

How do you know you are done?

- The test case compiles and links, but likely it crashes (you have not initialized anything, so this is not surprising)

Potential problems:

- You may sit and wonder what to do, when you should just make a stub. It's not that hard if you keep it simple. Maybe you don't need the whole interface as implied by the header. Let the linker error be your guide. Phone a friend if you are stuck.

Initialize Needed Data to Get the TEST to Run Without Crashing

Now you are pretty much on your own. Don't get discouraged. Get some help. Use the debugger.

When you finally have some meaningful tests, remember to refactor your tests and name everything appropriately.