(a simple Unit Test Framework for Embedded C)

Unity is a unit test framework. Our goal has been to keep is small and functional. The core Unity test framework is a single C and a couple header files, which provide functions and macros to make testing easier. Most of the framework is a variety of assertions which are meant to be placed in tests to verify that variables and return values contain the information that you believe they should. There are some additional methods for general test flow control.

We've developed Unity to be fairly cross-platform. It uses ANSI C for the library itself, and beautiful cross-platform Ruby for all the optional add-on scripts. We've personally used Unity with GCC, IAR's Embedded C Compiler, Clang, Green Hills, and MS Visual Studio. It shouldn't be too much work to get it to work with something else.

When you download Unity, you're going to get some other goodies as well. Some of these might be missing if you've downloaded a version meant for an end-user instead of developing Unity itself. Let's look at the root directory for a hint as to what those goodies are:

- test – These are tests using Unity which actually test unity itself. How cool is that?
- src – This is where Unity, and some example helpers to expand Unity, live
- examples – This has examples of how to use Unity and it's scripts.
- extras – This contains non-core modules which provide extra functionality, for example a test fixture designed to make Unity act more like CppUTest.
- docs – This has our wonderfully entertaining documentation (like this file)
- build – Just ignore this. Temporary build files get thrown here.
- auto – This contains a collection of Ruby scripts which are going to make using Unity painless
- targets – This is full of yaml configuration files. They are primarily here for unity's self tests when running rake (see config[] options) but are also a good reference for options used.

You're also going to see a makefile and a rakefile (plus a rakefile_helper... which strangely enough just helps the rakefile).

The makefile is a simple makefile which can be used to get the Unity tests going... it's also a good example of a simple way of assembling your tests. It's currently written assuming you are using GNU Make, but is simple enough that you could tweak it to use with something else.

The rakefile does the same thing, but using Rake. If you have Ruby and Rake installed you can use Rake instead of Make. It's going to provide you with extra goodies like test summaries and the ability to automagically discover your test functions (so you don't have to remember to call each one by hand).