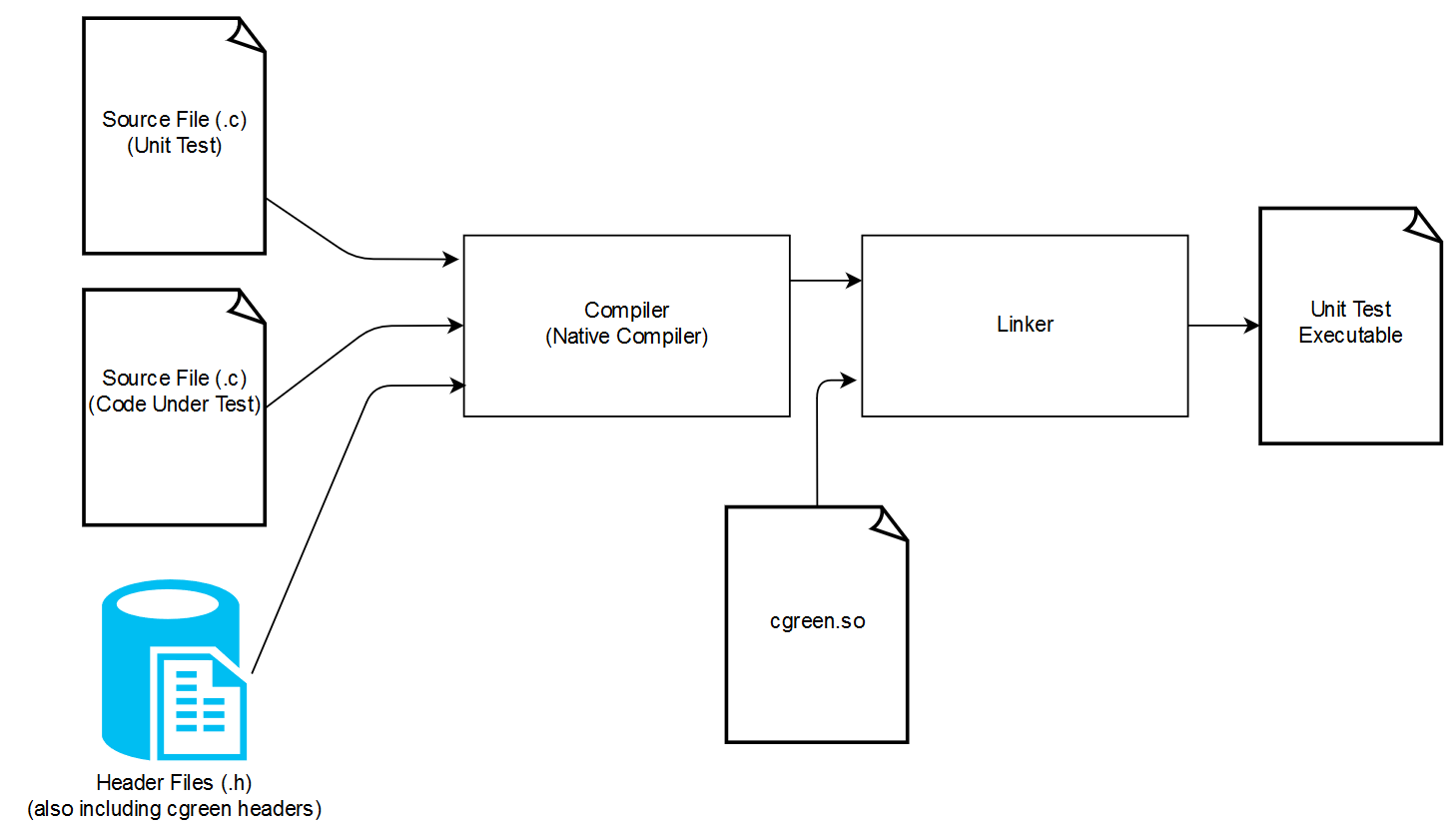
SCons - Building Unit Tests

Introduction

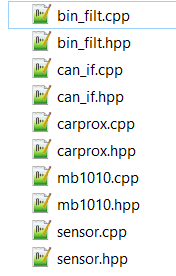
A unit test executable file shall be compiled using the following files:

1. Unit test source file (.c)
   1. This file contains test case definition and function calls to the Cgreen library
2. Code under test source file (.c)
   1. This file contains code to be executed by the ARM target; the objective is to test the logic of the code in this file
3. A bunch of header files (.h)
   1. Most of the header files are from the embedded project. These header files are used only for their symbols.
   2. The rest of the header files are from Cgreen
4. Cgreen library shared object file (.so)
   1. Required for load-time dynamic linking
   2. Contains Cgreen library function definitions



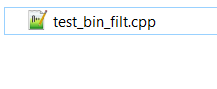
Unit Test - File Name Convention (not an official convention)

For example, the Sensors directory has the following source files in the image below. As a demonstration, I want to unit test bin\_filt.cpp, a data set filter module.



Files in foxp2/L5\_Application/FOXP2/Sensor

SCons is set up to look for the file test\_bin\_filt.cpp in the directory foxp2/test, where test\_ is prepended to the file of interest.



Files in foxp2/test/Sensor

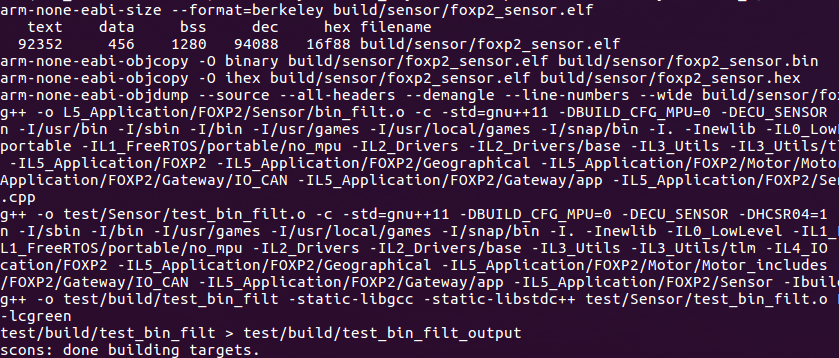
Note, all files prepended with test\_ located in the root directory foxp2/test or in a subdirectory will be built then executed automatically.

Unit Test - Building

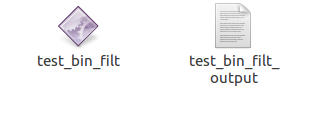
Invoke SCons and build an ECU project with the option --unit\_test



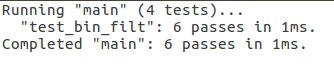
SCons will build then execute every unit test that exists within foxp2/test



SCons pipes the executable’s standard out to a new file with \_output appended to the name. The artifacts will be generated in foxp2/test/output.



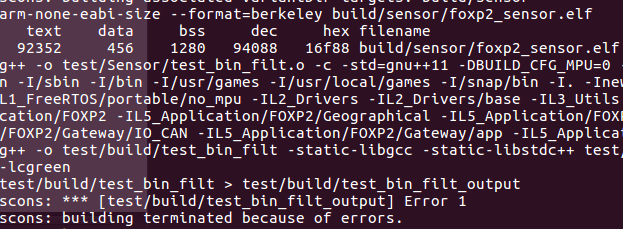
Files in foxp2/test/output

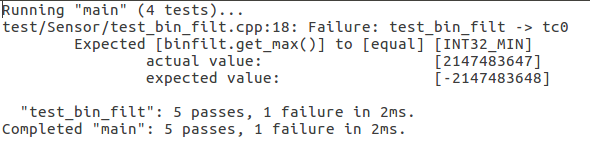


Content of test\_bin\_filt\_output

Unit Test - Building (continued)

If at least one unit test fails (error code is non-zero), the SCons build will fail.





Content of test\_bin\_filt\_output

Setup - Linux

Cgreen: <https://github.com/cgreen-devs/cgreen>

1. Git clone to get source code files
2. Follow instructions to build

* Or just run these commands:



1. Modify the environment variable LD\_LIBRARY\_PATH to point to the shared object’s directory



SCons



ARM toolchain



Setup - Linux (continued)

Sublime (optional)

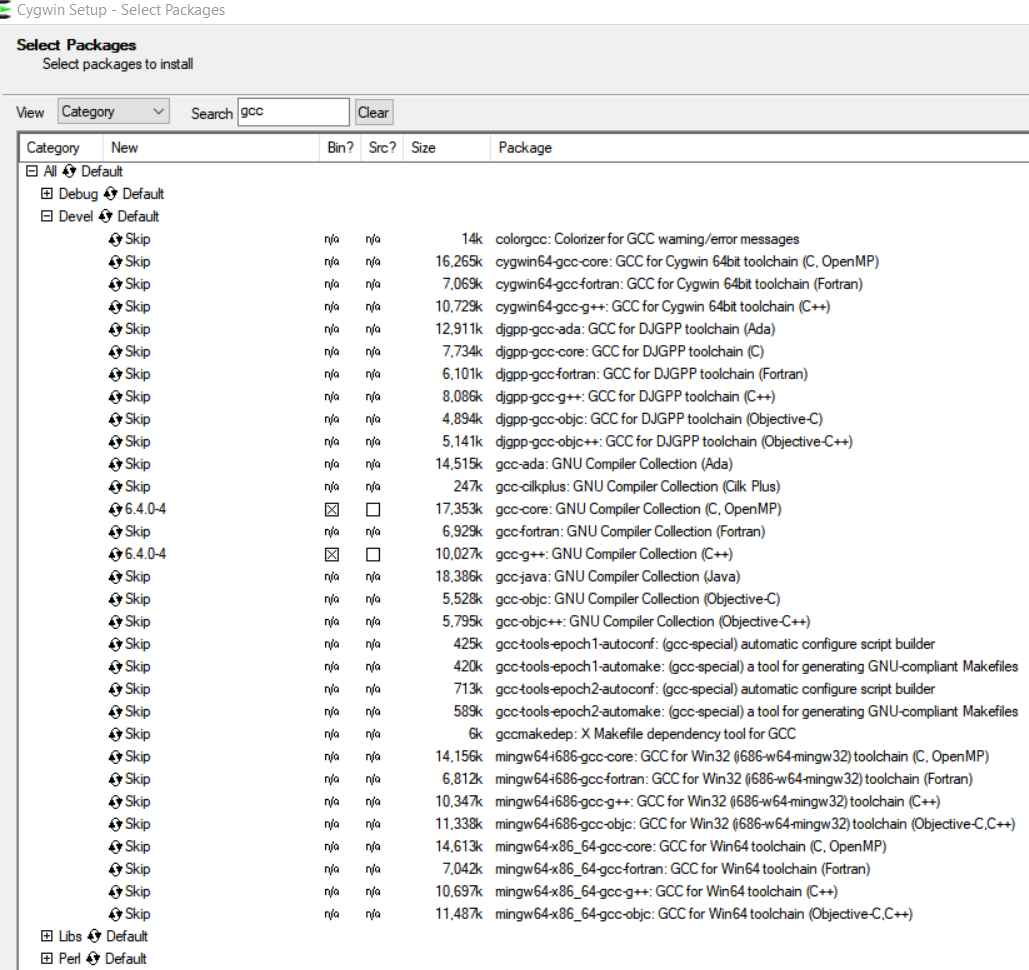


Setup - Windows

For Windows, I compiled Cgreen using Cygwin’s toolchain for x86 architecture; the produced linkable libraries are stored in the repository and are used during compile-time linking and load-time linking.

Cygwin Toolchain: <https://cygwin.com/install.html>

* Download the installer for x86 architecture (not x86\_64)
* During installation, install at the default destination: C:\cygwin
* During installation, you will need to install additional packages:
  + Devel – gcc-core: GNU Compiler Collection (C, OpenMP)
  + Devel – gcc-g++: GNU Compiler Collection (C++)



SCons: http://scons.org/