

DEVELOPING OOSIML AND C++ PROGRAMS

Using Codeblocks

on Linux

Dr. José M. Garrido
Department of Computer Science

Updated November 2016

College of Computing and Software Engineering
Kennesaw State University

© 2015, J. M. Garrido

1 Introduction

There are several Integrated Development Environments (IDEs) that help implementation of programs written in C, C++, Fortran, and other programming languages. Eclipse is one of the most complete and powerful tools, it is mainly useful for Java programs and in general, it can be very slow. For C++, C, and Fortran programs, Codeblocks and CodeLite are faster, lighter, and more convenient to use.

This document briefly explains implementing OOSimL programs using CodeBlocks on Linux and linking with the simulation library `oosimlib`. For a more detailed introduction to CodeBlocks, refer to various websites with CodeBlocks tutorial and documentation. For a simple but complete tutorial, visit the web page:

`ksuweb.kennesaw.edu/~jgarrido/comp_models`.

2 Preparing CodeBlocks

The following sequence of steps involve the basic procedure for setting the appropriate options on Codeblocks for using the OOSimL translator then the remaining steps for editing C++ programs, compiling, and linking with the simulation library.

2.1 Download the OOSimL Translator

The preliminary step required is the installation of the the translator and simulation library in the computer system running Linux. The 32-bit and the 64-bit versions of OOSimL translator are available on the web page mentioned previously. Use a package manager such as Synaptic, Ubuntu Software Center, or other to install the GSL.

1. Download the archive file `SCL_soft_models.tar.gz` from the web page and extract all files from archive.
2. Select the executable file of the SCL translator (`scl.out` or `scl64.out` depending on the version of Linux your computer (32-bit or 64-bit). Rename `scl64.out` to `scl.out` in a 64-bit Linux. This executable file, the `scl` script file and the `scl.h` file must be stored on a folder such as `~/SCL`.

2.2 Setting the Codeblocks Options

1. Start CodeBlocks, the screen that appears is shown in Figure 1. Click on 'Create a new project'.

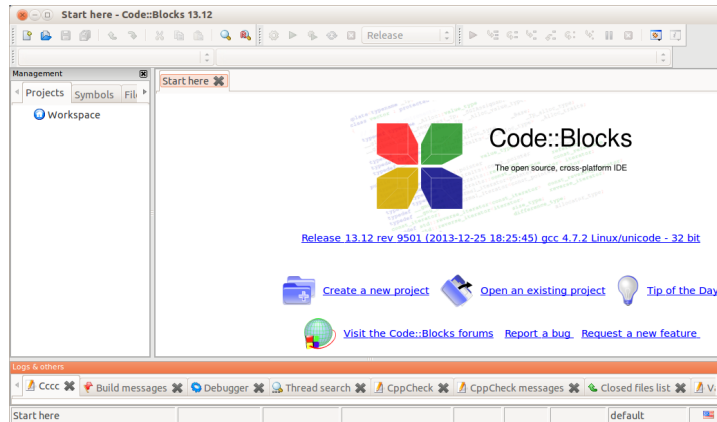


Figure 1: Starting Codeblocks.

2. On the top bar click on the Settings menu, then select Compiler. A dialog box appears. On the tab group, activate the Other Options tab, and click the button Advanced options located on the lower right of the dialog box. See Figure 2.

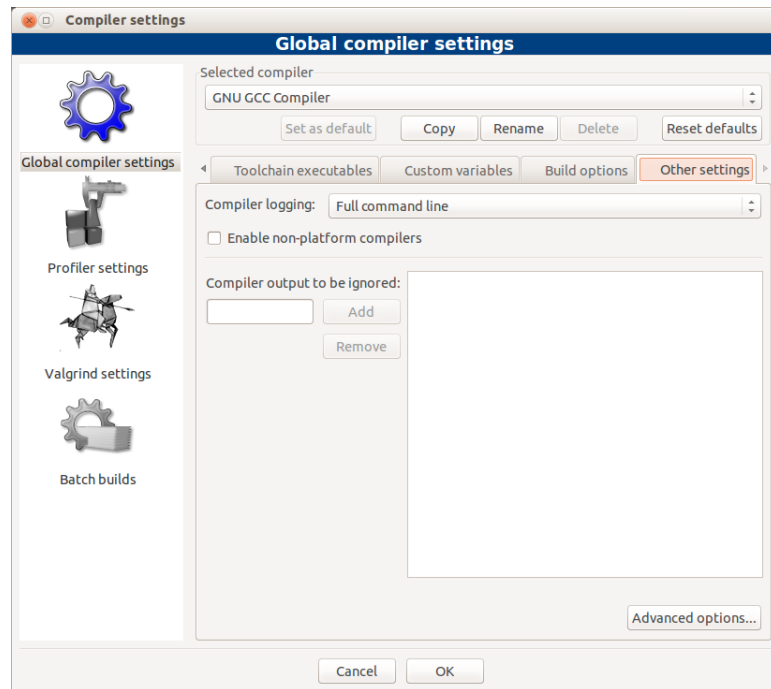


Figure 2: Selecting Advanced options.

3. Click the Yes button when the box *Edit advanced compiler setting?* appears.
4. On the new window that appears, click the '+' button and type `sc1` in the dialog box.
5. Type the Command line macro and the Generated files. The window will now appear as shown in Figure 3. This assumes that the SCL translator (executable file `sc1`) is located in folder: `~/SCL`

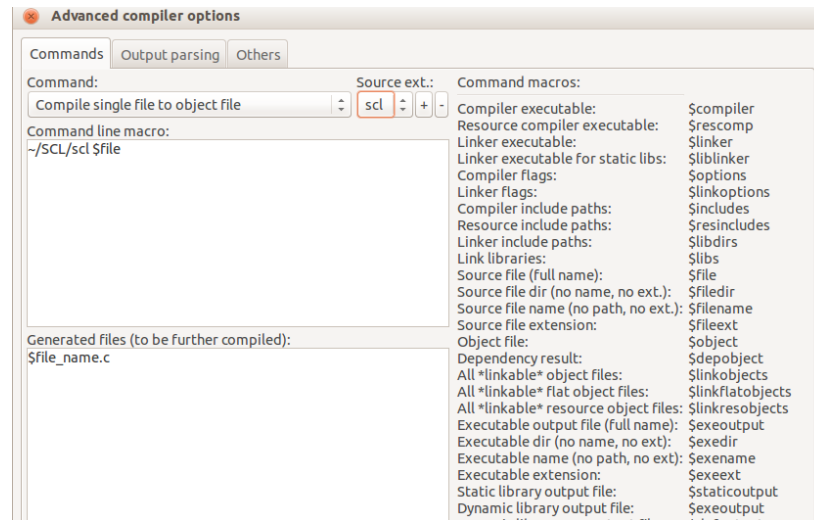


Figure 3: Adding the SCL extension and command for translating SCL files.

6. Click the Ok button located on the bottom of the window.

Codeblocks is now setup to recognize source files with an `scl` extension for editing and invoking the SCL translator.

2.3 Using Codeblocks for Implementing SCL Programs

1. Select Console Application and click the Go button, which is located on the upper right corner of the window.
2. On the Console Application window, click the Next button. Select C language and click Next.
3. Type the project title (name), e.g. *Welcome*. In this example, the project will be created in folder: `/home/jgarrido/comp_models/scl_models`, as shown in Figure 4. Click the Next button.

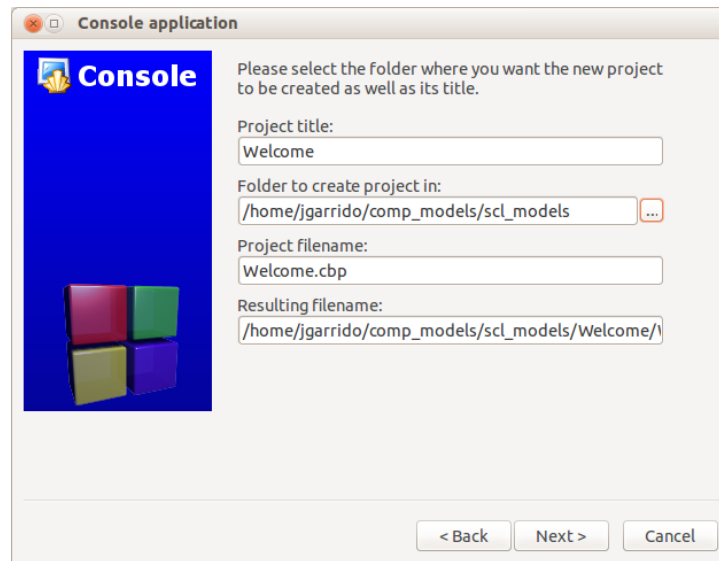


Figure 4: Title and location of project.

4. Select GNU GCC Compiler and click the Finish button.
5. Activate the left pane of the screen (Management), click the Projects tab. Remove the C source file main.c by right-clicking on it. The file appears on the editor pane. Now you can edit this C source file.
6. A new source file can be created by selecting File menu, then New, and Empty file. Click Yes to add this empty to the project. A new dialog window appears, type the name of the file with its `scl` extension. Now you can start editing this source file and when finished, save the file.
7. If one or more existing source files are to be included in the project, select the Project menu on the top bar, or right-click on the project name. Select Add files. Select the directory of the source to add to the project and select the SCL source file(s). Click Open, see Figure 5.

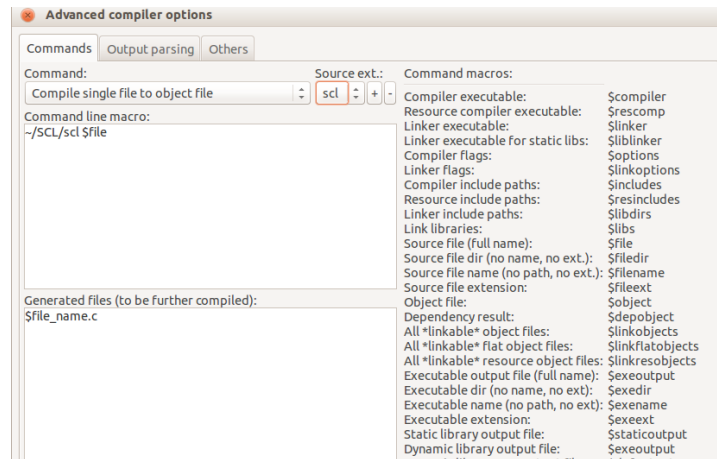


Figure 5: Adding an existing SCL source file to a project.

8. The source files are now under Others and under the project name. Double-click on the desired source file to edit it further. The file now appears on the edit area of the screen, as shown in Figure 6.

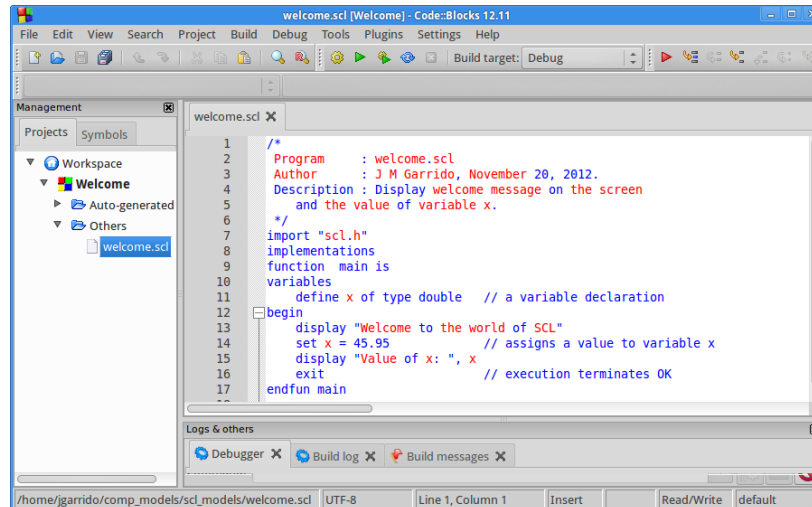


Figure 6: Editing an SCL program.

9. Right-click on the current project name, or activate the Project menu in the top bar, and select Build options.

- (a) On the tab Compiler settings, check Enable all compiler warnings.
- (b) This is an optional step, for small programs may not be required. For computational models, at least the GSL library may need to be added. On the tab Linker settings, click the Add button to add a library to the project. Repeat to add all necessary object files and libraries. For example, for a model `Plineq_sol3`, a local library `basic_lib`, two GSL sub-libraries are added `gsl` and `gslcblas`, and the `m` (standard math library). This is shown in Figure 7.

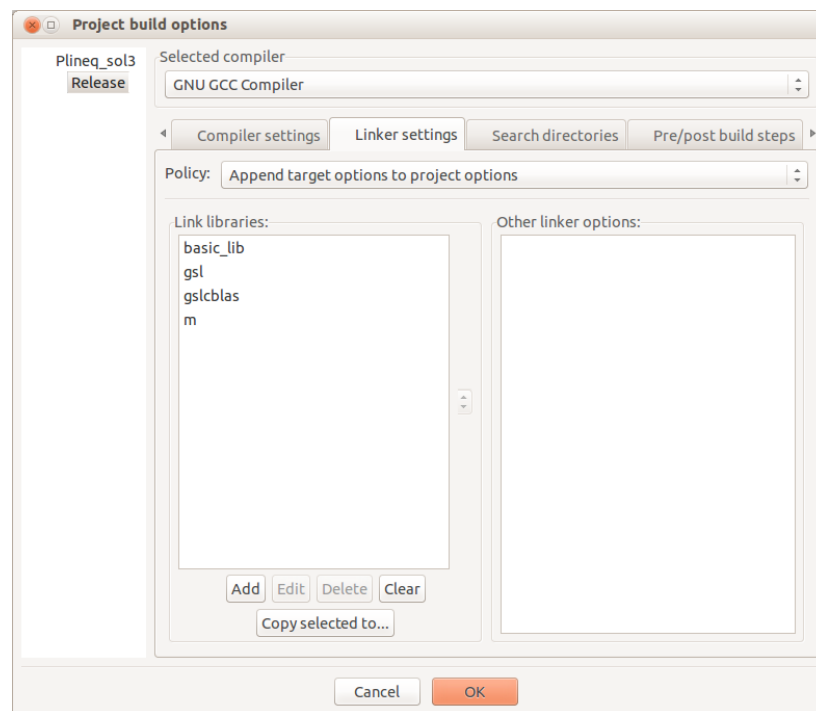


Figure 7: Linker settings.

- (c) On the tab Search directories and the tab 'Compiler', add the search directory for header files required by the source program while compiling. In Figure 8, the header file required by the program is located in the folder `~/SCL`.

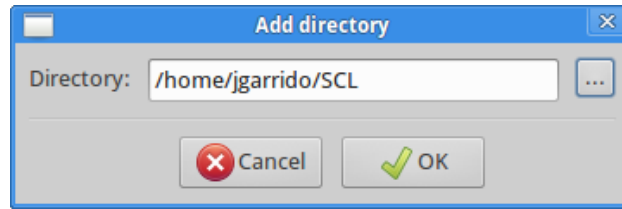


Figure 8: Adding a compiler search directory.

- (d) On the tab Search directories and the tab 'Linker', add the search directory of the libraries if needed. On Linux Ubuntu, the external libraries are always stored in standard system directories. The local library *libbasic_lib* library can be stored in any location, for example `~/comp_models/basic`. For this, click the Add button, then click (...) and navigate to the appropriate directory.
10. Build the project, which translates the SCL source file, compiles, and links the files in the project. On the top bar, select the Build menu and select the Build option. The Build log appears in the lower pane of the Codeblocks screen and is shown in Figure 9.

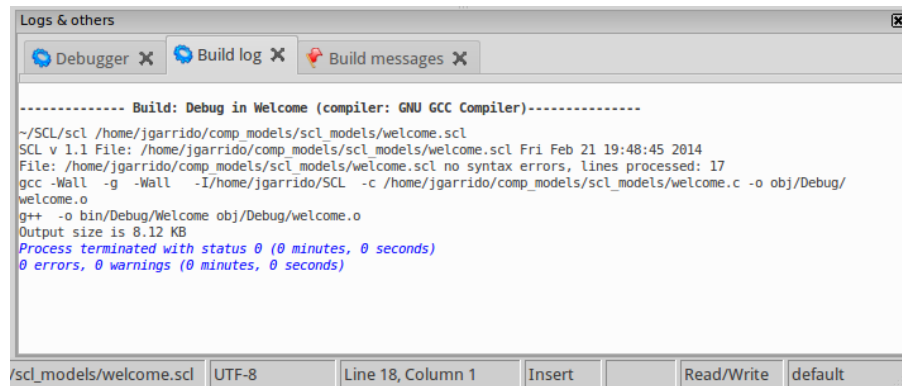


Figure 9: Building the project.

11. To execute the program, select the Run option in the Build menu. A new screen appears with the results of the execution, as shown in Figure 10. After the program terminates execution, press the Enter key.

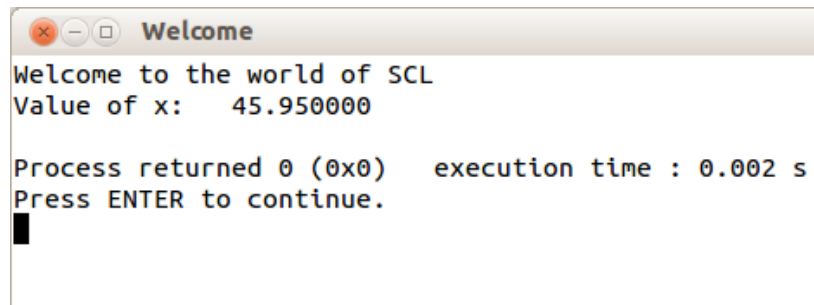


Figure 10: Executing the project.

12. On the top bar, activate the File menu and select Close project. Because the project was created in the directory `~/comp_models/scl_models`, Codeblocks creates several new directories and the executable file is located in the directory `~/comp_models/scl_models/Welcome/bin/Debug`.