Wiring up `deal.II` in Xcode

Oliver Sutton

January 23, 2018

These notes explain how to set up a new Xcode project to compile against the `deal.II` library, and are written with macOS version 10.12, Xcode version 9.2 and `deal.II` version 8.5 in mind. Note that there may be better ways to do some of these things which I haven't worked out yet.

1 Start a new Xcode project

Open Xcode and go to File → New → Project. Select the 'macOS' tab at the top and double click on 'Command Line Tool', as shown in Figure 1. Pick a name for the project, set 'Organization Identifier' to some value unique to you, and pick 'C++' for the language. After clicking next, you will be asked where the root directory of the project should be placed, so navigate to some appropriate place on the computer. At this point you can also choose whether you want to create a git repository for your code[1].

2 Navigating Xcode

Now you're in the Xcode project! Some of the different components are described in Figure 2.

Click on the `main.cpp` file and you should see a sample piece of code (shown in Figure 3) which just prints out `Hello, World!`. Click the run button in the top left hand corner of the window and the output console will pop up from the bottom and print this out.

You can add additional source files (i.e. `name.h` and `name.cpp`) files by going through File → New → File, and selecting 'C++ File'. It will offer to automatically create a corresponding header file for you.

3 Setting it up to work with `deal.II`

Find out where the `deal.II` files are installed to. If you have installed `deal.II` using the App, then this will be in `/Applications/deal.II-8.5-brew.app/Contents/Resources`, and if you compiled it from source yourself then it will be wherever you installed it to, and some of the paths below[2] will be different. From now on, I will refer to this directory as `DEALII_ROOT`.

In the Xcode file list, open the blue project file (this should be at the top of the list), select the 'Build Settings' tab, opt to view 'All' settings, and search for the key phrase 'search', as shown in pink in Figure 4.

Also shown in Figure 4 are the settings we will need to change here, marked A, B, and C. Double click on each one in turn, and in the popup box which appears, enter the following paths, placing each one on a separate line using the '+' button, and remembering to expand `DEALII_ROOT`:

A: `DEALII_ROOT/lib/`
`DEALII_ROOT/brew/lib/`
`DEALII_ROOT/brew/Cellar/tbb/2017_U5/lib/`

B: `DEALII_ROOT/include/`
`DEALII_ROOT/brew/include/`

[1] I find it useful to, so that you can keep track of what you've changed, but this is entirely optional.
[2] Specifically, those including the `brew` directory.

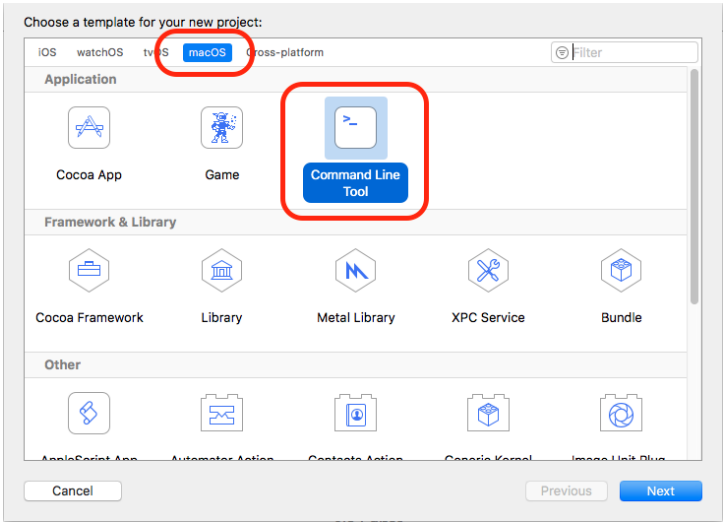

Figure 1: Creating a new C++ Xcode project

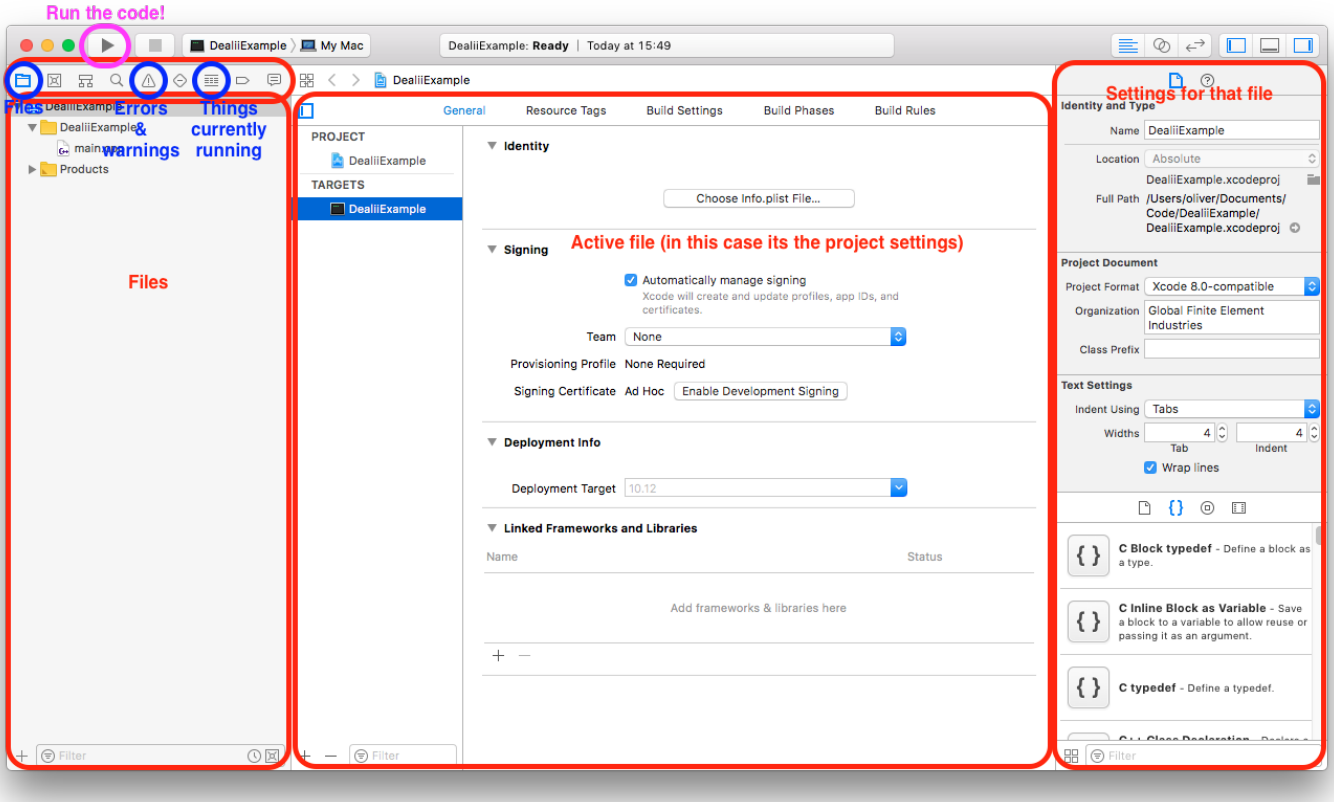

Figure 2: The basic buttons of Xcode

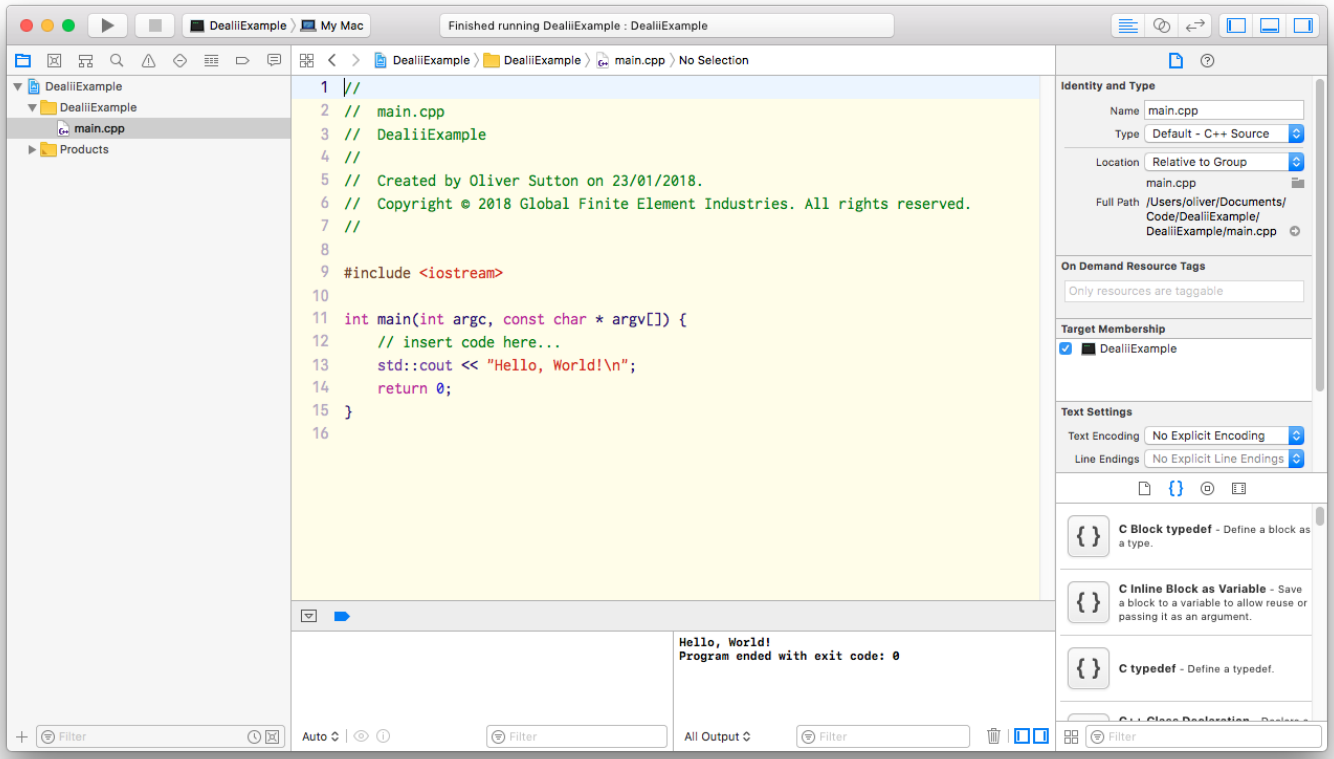

Figure 3: The starting code

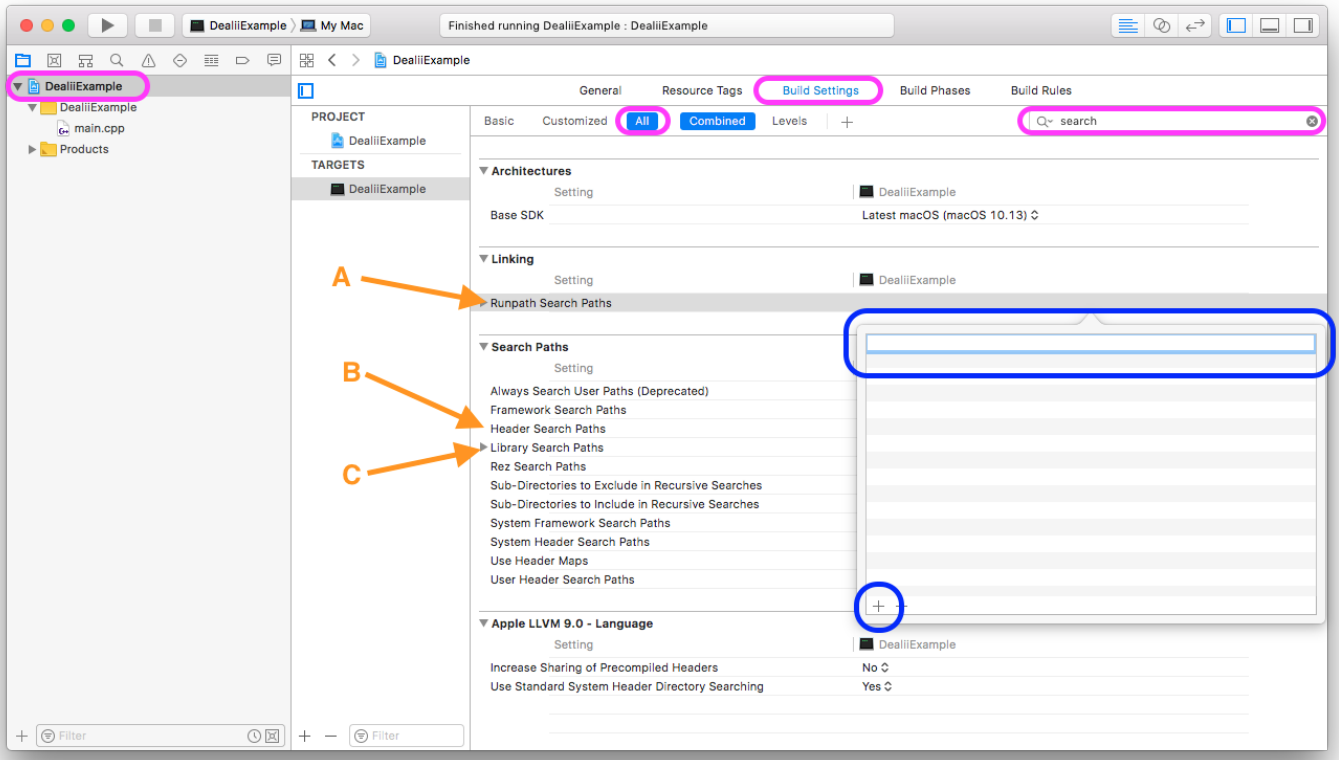

Figure 4: Setting the Xcode project settings

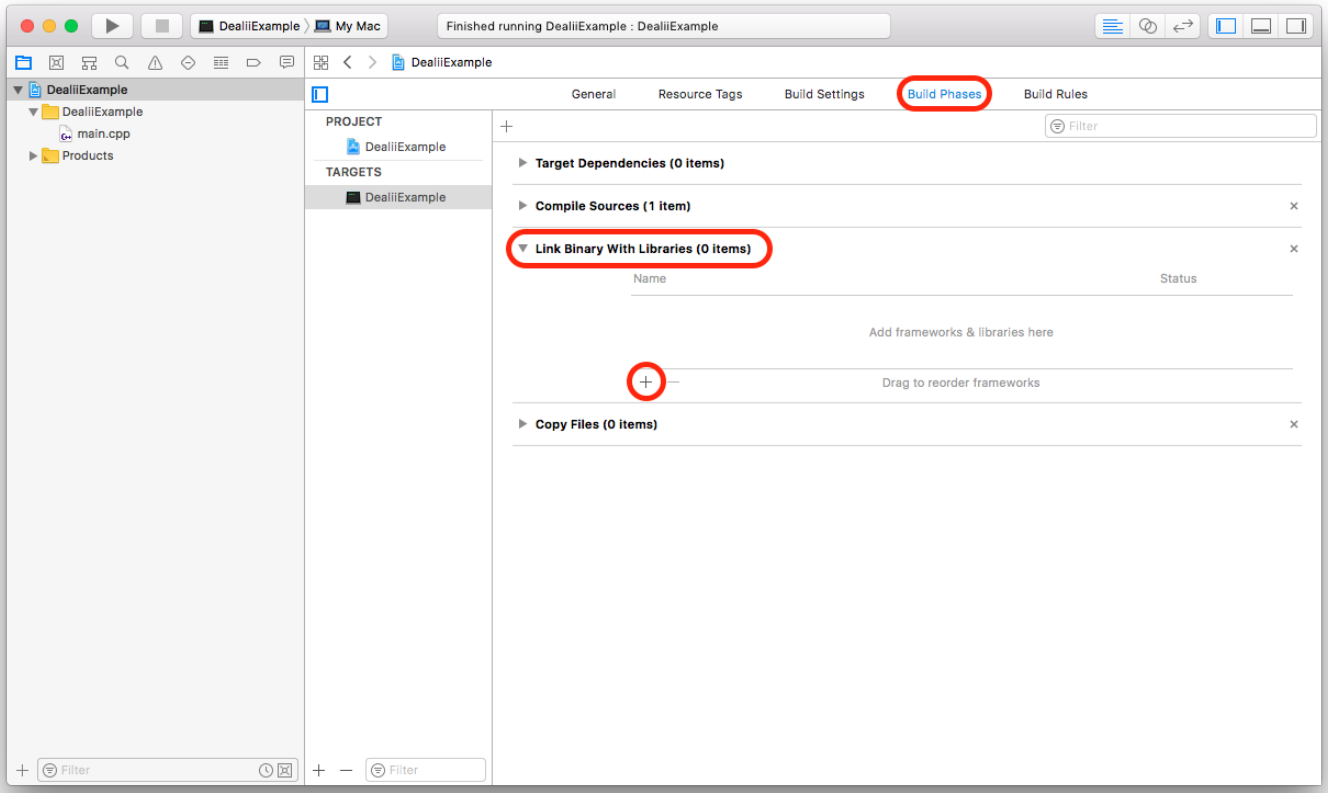

Figure 5: Telling Xcode to link the code to the libraries

C: `DEALII_ROOT/lib/`
`DEALII_ROOT/brew/lib/`

Now select the 'Build Phases' tab at the top, and open the 'Link Binary With Libraries' section, as shown in Figure 5. Click the '+' button to add a new library, and click 'Add Other' at the bottom of the popup. Now press `Cmd + Shift + G` to go to a specific directory, type in the path `DEALII_ROOT/lib/` and press enter. Select the file named `libdeal_II.dylib` and press 'Open'. Click the '+' button again and repeat, this time with the path `DEALII_ROOT/brew/Cellar/tbb/2017_U5/lib/`, and select the file named `libtbb.dylib`.

4 Checking it works

Now everything should be ready to go. Open one of the `deal.II` example steps, and copy and paste the code into the main file, replacing all of its contents. Click run, and see whether things work. A red hexagon with an exclamation mark indicates that a compiler error has occurred (so the code can't be run), while a yellow triangle with an exclamation mark denotes a warning (so the code will still run, but there is potentially something funny going on).

5 A note on output

By default, the `deal.II` example programs save any output files to the same directory that contains the *executable* produced by the compiler. In order to prevent your code directory getting dirty, Xcode places this executable in a well-hidden temporary directory. There are therefore two ways to find your output files:

1. Change the code to ensure that output is saved to a more sensible location. This is usually achieved by

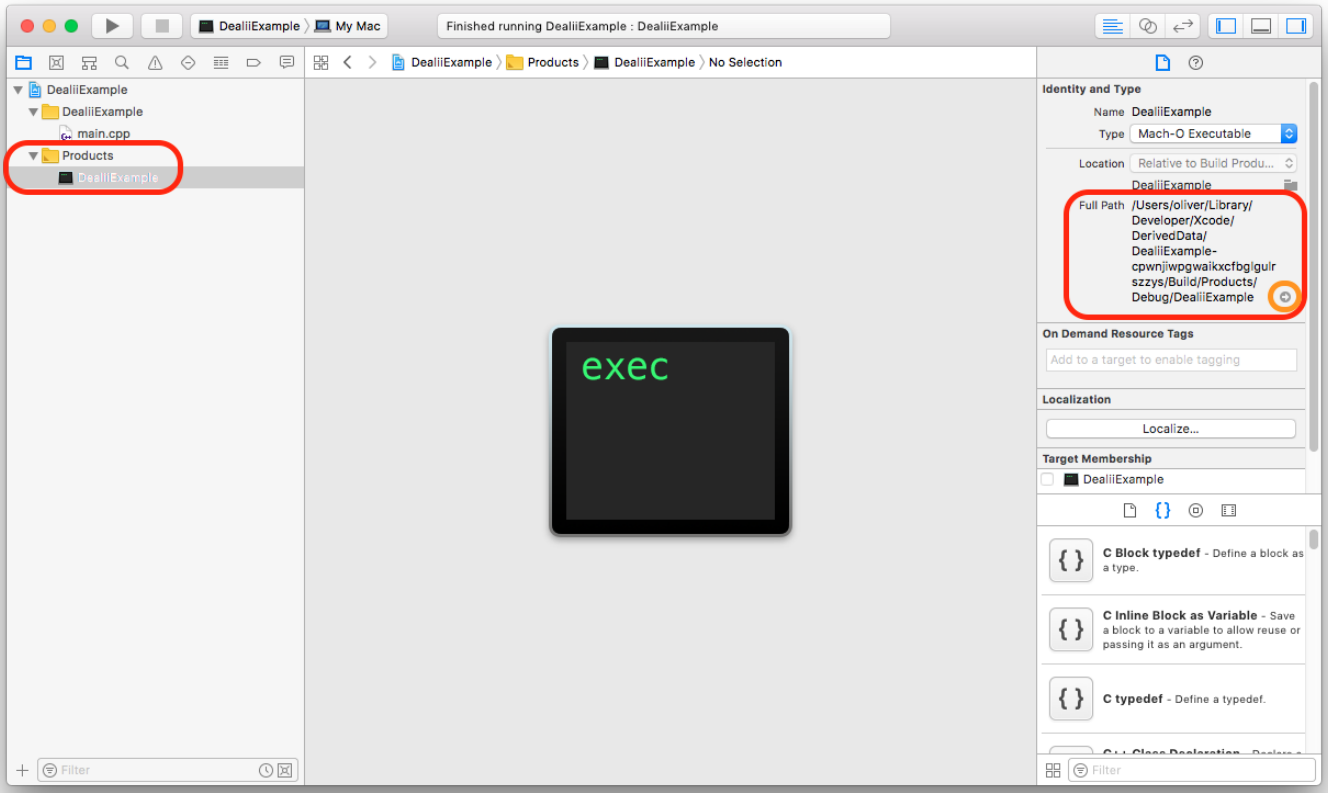

Figure 6: Finding the Xcode Derived Data directory

modifying the `output_results` function to use a filename like `/Users/USER_NAME/Data/solution`, where `USER_NAME` is your user name.

2. Navigate to the Xcode Derived Data directory where the executable is placed. This can be found by opening the 'Products' group in the file list, and selecting the executable file it contains. In the right-hand panel, click on the arrow in the circle next to 'Full Path'. This will open a Finder window showing the directory and any files it contains. This is shown in Figure 6