

Instructions for Installing Pyroms

The following version of Pyroms uses CMake to build the many different external libraries that are required for Pyroms to work. This means that you will want to download and install a version of **CMake 2.8** or better on your system before attempting to go any further in the build. Once CMake is installed, follow the instructions below for building the code:

1. Create a new directory in which to create an out-of-source build of the Pyroms software (ex. mkdir outofsource)
2. `cd outofsource/`
3. At this point, you can either use `ccmake` (a command line tool) or the CMake GUI to create the out-of-source build of Pyroms. This example will use `ccmake`, but the steps are identical in both. Enter the command: `ccmake ../pyroms` (or enter the path to your untarred pyroms directory if this is not one directory level up)
4. When the screen opens, press 'c' to create an initial configuration for your system.
5. This initial configuration will need a few things modified to find all of the shared libraries and include files this build requires. The system automatically attempts to find the netcdf shared libraries along with the hdf5 shared libraries by going through your `LD_LIBRARY_PATH` or `DYLD_LIBRARY_PATH`. You need not have hdf5 libraries available if the netcdf shared library was not built with hdf5 enabled. The first time you configure, it will show you whether or not it found those libraries (ex. Netcdf Shared Library Found : True). If this does not find the netcdf library or it does not find the hdf5 library and you build netcdf with hdf5 enabled, you will need to modify your `LD_LIBRARY_PATH` and try running the configure again.
6. `Netcdf.inc` must also be included in the build. This is searched for in common locations such as `/usr/local/include/` but if it cannot be found, you must create an environment variable called `NETCDFINC`, which will be the absolute path to the directory containing `netcdf.inc`. Re-run the configuration with the new environment variable and `netcdf.inc` should be included. You can also simply modify the field in `ccmake` called `Netcdf_INC` with the absolute path to the directory containing `netcdf.inc` instead if you would prefer.
7. There are two more options that must be modified for the build to be created properly. You must modify the two fields called: `EXECUTABLE_OUTPUT_PATH` and `LIBRARY_OUTPUT_PATH` to be locations where you would like to store the executables and library objects that are being created as part of the build. The build system will also search these for libraries that have been created when they are used as dependencies for other portions of the build, so setting these is critical for the build to run. You can set these to be your usual locations for bin & lib respectively, or you can define a completely different location to store these files. (ex. `EXECUTABLE_OUTPUT_PATH = /usr/local/bin`)

8. When all of the changes above have been made, press ‘c’ once more to configure all of these changes into the build.

9. Press ‘g’ to generate the Makefile used to build the software.

10. Run ‘make’ and this will go through the process of building the pyroms package along with its included dependencies. When make is done, it will have installed the python code into the path: `$LIBRARY_OUTPUT_PATH/python<version#>/site-packages/`. You will need to move these files to where your PYTHONPATH environment variable points now after finishing the rest of the steps in the build, or append this location to your PYTHONPATH.

11. Run “make scrip.so” or “make scrip.dylib” depending on your system. The resulting shared library object will be placed into `$LIBRARY_OUTPUT_PATH`.

12. Run “make pyroms_toolbox”. The resulting directory will be placed into your `$LIBRARY_OUTPUT_PATH`.

13. Copy `pyroms_toolbox` from your `${LIBRARY_OUTPUT_PATH}` to `${LIBRARY_OUTPUT_PATH}/python<version#>/site-packages/`. To use `pyroms_toolbox` in the Python interpreter, simply type: `import pyroms_toolbox`

14. Copy `libgridgen.so` (.dylib on MacOSX) to the installed pyroms directory in site-packages, which unless you’ve moved your build is done like: `(cp libgridgen.so $LIBRARY_OUTPUT_PATH/python2.6/site-packages/pyroms/)`
`LIBRARY_OUTPUT_PATH` is a placeholder for the absolute path that is located in the `LIBRARY_OUTPUT_PATH` variable, use the absolute path in place of `LIBRARY_OUTPUT_PATH`.

15. Copy `scrip.so` (.dylib on MacOSX) to the installed `pyroms/remapping/` directory, which is very similar to the above process except `scrip.so` must go into the sub- directory `remapping` inside of the `pyroms` package. (Example: `cp scrip.so $LIBRARY_OUTPUT_PATH/python2.6/site-packages/pyroms/remapping/`) Again, do not use `LIBRARY_OUTPUT_PATH`, use the absolute path that is contained in `LIBRARY_OUTPUT_PATH` for your build.