Build issues for Screamer V4.2.6 Mac OS X High Sierra, Windows 10, and Ubuntu LINUX

On the Mac, make sure you have the latest version of Xcode and the Command Line Tools installed (Xcode V9.2 as of 2018-01-17). Check with the macOS App Store.

Always start with the latest version of gcc gfortran (7.1) and HPC gcc (7.1) and Apple (8.0.0) clang compilers. Apple no longer provides the gcc compilers. You can find out what you are using by typing the below.

Go to http://hpc.sourceforge.net/index.php and download the binaries for gcc and gFortran. Follow the installation instructions.

Go to https://developer.apple.com/downloads/ and download the latest tools and/or xcode

NOTE: gFortran defaults to 32 bits for Windows and LINUX. I determined that it is necessary to include the following option in the gFortran call — add " -mcmmodel=medium". As of 2015-03-22 I have included this in the default .bat file for all 64 bit versions.

NOTE: 64 bit versions are required for Screamer V4.+ due to the memory requirements.

With XCode versions greater than 4 you will need to download the command-line tools as an additional step. You will find the option to download the command-line tools in XCode's Preferences. On 10.9 Mavericks or higher, you can get the command-line tools by simply typing xcode-select --install. I prefer to go to Apple Developer’s Web Site and download the command line pkg directly.

After HPC binaries and xcode command-line tools are installed, this is what you see.

As of 2018-01-17

gcc gfortran
MacBook-Pro-RBS:Downloads rbspielman$ gfortran -v
Using built-in specs.
COLLECT_GCC=gfortran
COLLECT_LTO_WRAPPER=/usr/local/libexec/gcc/x86_64-apple-darwin15.6.0/7.1.0/lto-wrapper
Target: x86_64-apple-darwin15.6.0
Configured with: ../gcc-7.1.0/configure --enable-languages=c++,fortran
Thread model: posix
gcc version 7.1.0 (GCC)

hpc version of g++
MacBook-Pro-RBS:Downloads rbspielman$ g++ -v
Using built-in specs.
COLLECT_GCC=g++
COLLECT_LTO_WRAPPER=/usr/local/libexec/gcc/x86_64-apple-darwin15.6.0/7.1.0/lto-wrapper
Target: x86_64-apple-darwin15.6.0
Configured with: ../gcc-7.1.0/configure --enable-languages=c++,fortran
Thread model: posix
gcc version 7.1.0 (GCC)

Apple Clang
MacBook-Pro-RBS:Downloads rbspielman$ clang -v
Apple LLVM version 9.0.0 (clang-900.0.39.2)
Target: x86_64-apple-darwin17.3.0
Thread model: posix
InstalledDir: /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault.xctoolchain/usr/bin

Note: As of Yosemite Apple points back to the gcc compiler with g++ not clang!
-------------------------------------------------------------------------------------
NOTE: in /usr/lib/ there are three versions of libstdc++. They are: libstdc++.dylib and libstdc++.6.dylib. It is not clear which is the correct library to use. It is not clear which is the DEFAULT library used.
-------------------------------------------------------------------------------------
NOTE: in /usr/local/lib/gcc/ there are multiple versions of x86_64-apple-darwin. We use x86_64-apple-darwin15.6.0.
-------------------------------------------------------------------------------------
Build #1 Force use of the gcc g++ linker; use the default gcc gfortran compiler; and use the default library calls.

rm *.o *.a
gfortran -c -O04 *.f
ar crv screamer.a *.o
rm *.o
ranlib screamer.a
/usr/local/bin/g++ -c cprog.cpp
/usr/local/bin/g++ -o screamer cprog.o screamer.a -L/usr/local/lib -lgfortran

The calls to /usr/local/bin/ force the use of the latest HPC version of c++. (Just in case the default is not the latest version.) When you do this you get the following dylibs. There are three dylib’s that are located in /usr/local/lib/ (shown in the otools call below) that will not be on Macs that do not have the gcc compilers added. This gives a dynamic build that requires any Mac trying to run THIS binary to have all of these dynamic libraries installed. (For Yosemite and later the call to /usr/local/bin/ is the same as the default g++ call.)

With Screamer V4.1.3 and earlier, the build works with max_nodes = 8191 but fails with max_nodes = 8192 - go figure.

In Yosemite and later you get the same result with:

rm *.o *.a
gfortran -c -O04 *.f
ar crv screamer.a *.o
rm *.o
ranlib screamer.a
g++ -c cprog.cpp
g++ -o screamer cprog.o screamer.a -L/usr/local/lib -lgfortran

This dynamic compile works on my MacBook Pro with Yosemite OS X 10.11.2.

Ricks-MacBook-Pro:src rbspielman$ otool -L screamer
screamer:
    /usr/local/lib/libgfortran.3.dylib (compatibility version 4.0.0, current version 4.0.0)
Build #2 Fully dynamic but using Apple clang

Use the apple clang linker, gcc gfortran, and the default library calls. These are the same link options as Built #1.

```
rm *.o *.a
gfortran -c -O04 *.f
ar crv screamer.a *.o
rm *.o
ranlib screamer.a
clang -c cprog.cpp
clang -o screamer cprog.o screamer.a -L/usr/local/lib -lgfortran
```

**This build fails! Earlier versions of clang did work.**

Build #3 Use the gcc c++ and point to the default Apple c++ libraries in /usr/lib/

Use the default gcc c++ linker and the default gcc gfortran. Point directly to the default installed dylibs.

```
rm *.o *.a
gfortran -c -O04 *.f
ar crv screamer.a *.o
rm *.o
ranlib screamer.a
g++ -c cprog.cpp
g++ -o screamer cprog.o screamer.a /usr/lib/libstdc++.6.dylib /usr/local/lib/libgfortran.dylib
```

For reasons that are not clear otools below shows both the gcc version of the c++ dynamic library /usr/local/lib/libstdc++.6.dylib and the default dynamic Apple c++
dynamic library /usr/lib/libstdc++.6.dylib. Our goal was to eliminate the gcc c++
dynamic library in /usr/local/lib/, which is not usually installed on most Macs.

Ricks-MacBook-Pro:src rbspielman$ otool -L screamer
screamer:
   /usr/lib/libstdc++.6.dylib (compatibility version 7.0.0, current version
   104.1.0)
   /usr/local/lib/libgfortran.3.dylib (compatibility version 4.0.0, current version
   4.0.0)
   /usr/local/lib/libstdc++.6.dylib (compatibility version 7.0.0, current version
   7.21.0)
   /usr/lib/libSystem.B.dylib (compatibility version 1.0.0, current version
   1226.10.1)
   /usr/local/lib/libgcc_s.1.dylib (compatibility version 1.0.0, current version
   1.0.0)
-----------------------------------------------------------------------------------------------------------------------------
Build #3b Use the gcc c++ and point to the default Apple c++ libraries in /usr/lib/

Use the gcc c++ linker and gcc gfortran. Point directly to the default installed
dylibs.

rm *.o *.a
gfortran -c -O04 *.f
ar crv screamer.a *.o
rm *.o
ranlib screamer.a
g++ -c cprog.cpp
g++ -o screamer cprog.o screamer.a /usr/lib/libstdc++.6.0.9.dylib /usr/local/lib/
libgfortran.dylib

For reasons that are not clear otools below shows both the gcc version of the c++
dynamic library /usr/local/lib/libstdc++.6.dylib and the default dynamic Apple c++
dynamic library /usr/lib/libstdc++.6.dylib. Our goal was to eliminate the gcc c++
dynamic library in /usr/local/lib/, which is not usually installed on most Macs.

With Screamer V4.1.3 and earlier, the build works with max_nodes = 8191 but
fails with max_nodes = 8192 - go figure.
Build #4 Same as Build #3 but the default c++ pointer is removed. This is NOT a fully static Screamer build as you would still need the libgfortran dylib! Is there a static 32 bit fortran library?

I am pointing directly to the gcc gFortran dylib that I plan to use

```
rm *.o *.a
gfortran -c -O04 *.f
ar crv screamer.a *.o
rm *.o
ranlib screamer.a
g++ -c cprog.cpp
```
```
g++ -o screamer cprog.o screamer.a /usr/local/lib/libgfortran.dylib
```

I removed the direct call to the c++ library and ended up again with a dynamic c++ library in /usr/local/lib/. I get the same dylibs as I did with build #1.

With Screamer V4.1.3 and earlier, this build works with max_nodes = 8191 but fails with max_nodes = 8192 - go figure.

```
Ricks-MacBook-Pro:src rbspielman$ otool -L screamer
screamer:
    /usr/lib/libstdc++.6.dylib (compatibility version 7.0.0, current version 104.1.0)
    /usr/local/lib/libgfortran.3.dylib (compatibility version 4.0.0, current version 4.0.0)
    /usr/local/lib/libstdc++.6.dylib (compatibility version 7.0.0, current version 7.21.0)
    /usr/lib/libSystem.B.dylib (compatibility version 1.0.0, current version 1226.10.1)
    /usr/local/lib/libgcc_s.1.dylib (compatibility version 1.0.0, current version 1.0.0)
```

Build #4 Same as Build #3 but the default c++ pointer is removed. This is NOT a fully static Screamer build as you would still need the libgfortran dylib! Is there a static 32 bit fortran library?
Again, this is different than seen in Mavericks where we got:
/usr/local/lib/libgfortran.3.dylib (compatibility version 4.0.0, current version 4.0.0)
/usr/lib/libc++.1.dylib (compatibility version 1.0.0, current version 120.0.0)
/usr/lib/libSystem.B.dylib (compatibility version 1.0.0, current version 1197.1.1)

Build #5 Same as Build #4 but pointing to the static gFortran libraries (libgfortran.a) with a 64-bit build (gcc g++)

```
gfortran -c -O02 -mmodel=medium *.f
ar crv screamer64.a *.o
rm *.o
ranlib screamer64.a

g++ -c cprog.cpp

g++ -o screamer64 cprog.o screamer64.a /usr/local/lib/libgfortran.a /usr/local/lib/libquadmath.a /usr/local/lib/gcc/x86_64-apple-darwin15.6.0/7.1.0/libgcc.a
```

Goes to the latest version of x86_64-apple-darwin15.6.0/7.1.0

```
Ricks-MacBook-Pro:src rbspielman$ otool -L screamer64
screamer64:
    /usr/local/lib/libstdc++.6.dylib (compatibility version 7.0.0, current version 7.23.0)
    /usr/lib/libSystem.B.dylib (compatibility version 1.0.0, current version 1238.60.2)
    /usr/local/lib/libgcc_s.1.dylib (compatibility version 1.0.0, current version 1.0.0)
```
This is still not a fully static build!

Where in Mavericks I got:

/usr/lib/libc++.1.dylib (compatibility version 1.0.0, current version 120.0.0)
/usr/lib/libSystem.B.dylib (compatibility version 1.0.0, current version 1197.1.1)

Comments on Static Builds:

You can get rid of libgcc_s by actually specifying -static-libgcc on your command-line

- static-libgfortran

On systems that provide libgfortran as a shared and a static library, this option forces the use of the static version. If no shared version of libgfortran was built when the compiler was configured, this option has no effect.

-dynamiclib -lgfortran -static-libgfortran \
/opt/local/lib/gcc47/libquadmath.a -static-libgcc

gfortran is composed of two main parts: the compiler, which creates the executable program from your code, and the library, which is used when you run your program afterwards. That explains why, if gfortran is installed in a non-standard directory, it may compile your code fine but the executable may fail with an error message like library not found. One way to avoid this (more ideas can be found on the binaries page) is to use the so-called "static linking", available with option -static gfortran then put the library code inside the program created, thus enabling it to run without the library present (like, on a computer where gfortran is not installed). Complete example is:

gfortran -static myfile.f -o program.exe