Compiling GCC 6 on macOS Sierra

Posted on September 22, 2016 by Sol

In this tutorial, I will show you how to compile from source and install the current stable version of GCC with Graphite loop optimizations on your macOS computer. The instructions from this tutorial were tested with Xcode 8 and Sierra (macOS 10.12).

Clang, the default compiler for macOS, supports only C, C++, Objective-C and Objective-C++. If you are interested in a modern Fortran compiler, e.g. you will need *gfortran* that comes with GCC. Another reason to have the latest stable version of GCC on your macOS is that it provides you with an alternative C and C++ compiler. Testing your code with two different compilers is always a good idea.

In order to compile GCC from sources you will need a working C++ compiler. In the remaining of this article I will assume that you have installed the Command Line Tools for Xcode. At the time of this writing Apple's Command Line Tools maps the *gcc* and *g++* to *clang* and *clang++*. If you don't have the Command Line Tools installed, open a Terminal and write:

```
1 xcode-select --install
```

which will guide through the installation process.

Let's start by downloading the last stable version of GCC from the GNU website, so go to: http://gcc.gnu.org/mirrors.html and download gcc-6.2.0.tar.bz2. I've saved the archive in my *Downloads* folder.

We will also need three other libraries for a successful build of gcc: mpc, mpfr and gmp. Use the above links and download the last versions for all of them: gmp-6.1.1.tar.bz2, mpc-1.0.3.tar.gz and mpfr-3.1.4.tar.bz2, also save them in your *Downloads* folder.

For enabling the Graphite loop optimizations you will need two extra libraries, go to ftp://gcc.gnu.org/pub/gcc/infrastructure/ and download *isl-0.16.1.tar.bz2*.

Extract the above five archives in your Downloads folder and open a Terminal window.

We will start by compiling the gmp library:

```
1 cd ~
2 cd Downloads
3 cd gmp*
```

Create a new folder named *build* in which the compiler will save the compiled library:

```
1 mkdir build && cd build
```

And now the fun part … write in your Terminal:

```
1 ../configure --prefix=/usr/local/gcc-6.2.0 --enable-cxx
```

If you see no error message we can actually compile the gmp library:

```
1 make -j 4
```

In a few minutes you will have a compiled gmp library. If you see no error message … congratulations, we are ready to install the library in the */usr/local/gcc-6.2.0* folder (you will need the administrator password for this):

```
1  sudo make install
```

We will do the same steps for MPFR now:

```
1  cd ..
2  cd ..
3  cd mpfr*
4  mkdir build && cd build
```

Configuration phase:

```
1  ../configure --prefix=/usr/local/gcc-6.2.0 --with-gmp=/usr/local/gcc-6.2.0
```

The second parameter will just inform the configure app that gmp is already installed in */usr/local/gcc-6.2.0*.

After the configure phase is finished, we can make and install the library:

```
1  make -j 4
2  sudo make install
```

Now, we are going to build MPC:

```
1  cd ..
2  cd ..
3  cd mpc*
4  mkdir build && cd build
5  ../configure --prefix=/usr/local/gcc-6.2.0 --with-gmp=/usr/local/gcc-6.2.0 --with-mpfr=/usr/local/gcc-6.2.0
6  make -j 4
7  sudo make install
```

At this time you should have finished to build and install the necessary prerequisites for GCC.

Next step is to build the library for the Graphite loop optimizations:

```
1  cd ..
2  cd ..
3  cd isl*
4  mkdir build && cd build
5  ../configure --prefix=/usr/local/gcc-6.2.0 --with-gmp-prefix=/usr/local/gcc-6.2.0
6  make -j 4
7  sudo make install
```

We are ready to compile GCC now. Be prepared that this could take more than one hour on some machines … Since I'm interested only in the C, C++ and Fortran compilers, this is the configure command I've used on my machine:

```
1  cd ..
2  cd ..
3  cd gcc*
4  mkdir build && cd build
5  ../configure --prefix=/usr/local/gcc-6.2.0 --enable-checking=release --with-gmp=/usr/local/gcc-6.2.0 --with-mpf
```

../configure --prefix=/usr/local/gcc-6.2.0 --enable-checking=release --with-gmp=/usr/local/gcc-6.2.0 --with-mpfr=/usr/local/gcc-6.2.0 --with-mpc=/usr/local/gcc-6.2.0 --enable-languages=c,c++,fortran --with-isl=/usr/local/gcc-6.2.0 --program-suffix=-6.2.0

The above command instructs the configure app where we have installed gmp, mpfr, mpc and isl; also it tells to add a prefix to all the resulting executable programs, so for example if you will invoke GCC 6.2.0 you will write *gcc-6.2.0*, the *gcc* command will invoke Apple's version of *clang*.

If you are interested in building more compilers available in the GCC collection modify the *–enable-languages* configure option.

And now, the final touches:

```
1  make -j 4
```

Grab a coffee, maybe a book, and wait … this should take approximately, depending on your computer configuration, an hour … or more … and about 4.4GB of your disk space for the build folder.

Install the compiled gcc in */usr/local/gcc-6.2.0*:

```
sudo make install
```

Now, you can keep the new compiler completely isolated from your Apple's gcc compiler and, when you need to use it, just modify your path by writing in Terminal:

```
export PATH=/usr/local/gcc-6.2.0/bin:$PATH
```

If you want to avoid writing the above command each time you open a Terminal, save the above command in the file *.bash_profile* from your Home folder.

You should be able to invoke any of the newly compiled compilers C, C++, Fortran …, invoking g++ is as simple as writing in your Terminal:

```
g++-6.2.0 test.cpp -o test
```

Remember to erase your *build* directories from Downloads if you want to recover some space.

Next, I'll show you how to check if the compiler was properly installed by compiling and running a few examples. GCC 6 uses by default the C++14 standard and C11 for the C coders, you should be able to compile any valid C++14 code directly. In your favorite text editor, copy and save this test program (I'll assume you will save the file in your Home directory):

```
//Program to test the C++ lambda syntax and initializer lists
#include <iostream>
#include <vector>

using namespace std;

int main()
{
  // Test lambda
  cout << [](int m, int n) { return m + n;} (2,4) << endl;

  // Test initializer lists and range based for loop
  vector<int> V({1,2,3});

  cout << "V =" << endl;
  for(auto e : V) {
    cout << e << endl;
  }

  return 0;
}
```

Compiling and running the above lambda example:

```
g++-6.2.0 tst_lambda.cpp -o tst_lambda
./tst_lambda
6
V =
1
2
3
```

If you see a bunch of warnings like:

```
/var/folders/g0/9dd92g0n2yv3bxh33p_tp__00000gn/T//ccbK6ICf.s:194:11: warning: section "__textcoal_nt" is depre
        .section __TEXT,__textcoal_nt,coalesced,pure_instructions
                 ^      ~~~~~~~~~~~~~
/var/folders/g0/9dd92g0n2yv3bxh33p_tp__00000gn/T//ccbK6ICf.s:194:11: note: change section name to "__text"
        .section __TEXT,__textcoal_nt,coalesced,pure_instructions
                 ^      ~~~~~~~~~~~~~
/var/folders/g0/9dd92g0n2yv3bxh33p_tp__00000gn/T//ccbK6ICf.s:395:11: warning: section "__textcoal_nt" is depre
        .section __TEXT,__textcoal_nt,coalesced,pure_instructions
                 ^      ~~~~~~~~~~~~~
/var/folders/g0/9dd92g0n2yv3bxh33p_tp__00000gn/T//ccbK6ICf.s:395:11: note: change section name to "__text"
        .section __TEXT,__textcoal_nt,coalesced,pure_instructions
                 ^      ~~~~~~~~~~~~~
```

```
/var/folders/g0/9dd92g0n2yv3bxh33p_tp__00000gn/T//ccbK6ICf.s:433:11: warning: section "__textcoal_nt" is depre
        .section __TEXT,__textcoal_nt,coalesced,pure_instructions
                 ^      ~~~~~~~~~~~~~
/var/folders/g0/9dd92g0n2yv3bxh33p_tp__00000gn/T//ccbK6ICf.s:433:11: note: change section name to "__text"
        .section __TEXT,__textcoal_nt,coalesced,pure_instructions
                 ^      ~~~~~~~~~~~~~
/var/folders/g0/9dd92g0n2yv3bxh33p_tp__00000gn/T//ccbK6ICf.s:654:11: warning: section "__textcoal_nt" is depre
        .section __TEXT,__textcoal_nt,coalesced,pure_instructions
                 ^      ~~~~~~~~~~~~~
/var/folders/g0/9dd92g0n2yv3bxh33p_tp__00000gn/T//ccbK6ICf.s:654:11: note: change section name to "__text"
        .section __TEXT,__textcoal_nt,coalesced,pure_instructions
                 ^      ~~~~~~~~~~~~~
```

you can safely ignore them, these are related to the assembly code generated by GCC and have nothing to do with your C++ code.

Unfortunately, at this time, there is no way in which to instruct GCC not to use *__textcoal_nt*, or silence the above warnings. A quick and dirty workaround is to filter the output of the compiler with something like:

```
g++-6.2.0 tst_lambda.cpp -o tst_lambda 2>&1 >/dev/null | grep -v -e '^/var/folders/*' -e '^[[:space:]]*\.sectio
```

We could also compile a C++ code that uses threads:

```
//Create a C++ thread from the main program

#include <iostream>
#include <thread>

//This function will be called from a thread
void call_from_thread() {
    std::cout << "Hello, World!" << std::endl;
}

int main() {
    //Launch a thread
    std::thread t1(call_from_thread);

    //Join the thread with the main thread
    t1.join();

    return 0;
}
```

Next, we present a simple C++ code that uses regular expressions to check if the input read from stdin is a floating point number:

```
//Uses a regex to check if the input is a floating point number

#include <iostream>
#include <regex>
#include <string>

using namespace std;

int main()
{
  string input;
  regex rr("((\\+|-)?[[:digit:]]+)(\\.(([[:digit:]]+)?))?((e|E)((\\+|-)?)[[:digit:]]+)?");
  //As long as the input is correct ask for another number
  while(true)
  {
    cout<<"Give me a real number!"<<endl;
    cin>>input;
    if(!cin) break;
    //Exit when the user inputs q
    if(input=="q")
      break;
    if(regex_match(input,rr))
      cout<<"float"<<endl;
    else
    {
      cout<<"Invalid input"<<endl;
    }
```

```
28    }
29  }
```

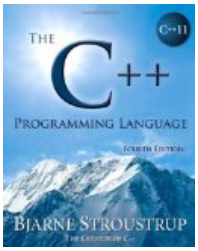
If you are a Fortran programmer, you can use some of the Fortran 2008 features like *do concurrent* with gfortran-6.2.0:

```
1  integer,parameter::mm=100000
2  real::a(mm), b(mm)
3  real::fact=0.5
4
5  ! initialize the arrays
6  ! ...
7
8  do concurrent (i = 1 : mm)
9      a(i) = a(i) + b(i)
10 enddo
11
12 end
```

The above code can be compiled with (assuming you've named it tst_concurrent_do.f90):

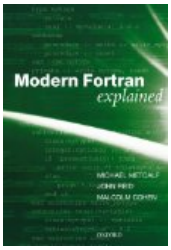
```
1  gfortran-6.2.0 tst_concurrent_do.f90 -o tst_concurrent_do
2  ./tst_concurrent_do
```

If you are interested in learning more about the new C++11/C++14 syntax I would recommend reading The C++ Programming Language by Bjarne Stroustrup.



or, Professional C++ by M. Gregoire, N. A. Solter, S. J. Kleper 2nd edition:

If you need to brush your Fortran knowledge a good book is Modern Fortran Explained by M. Metcalf, J. Reid and M. Cohen:



/Library/Developer/CommandLineTools", it should get rid of those errors (At least in xcode8

1 · Reply · Share ›

Kajika ➚ Solian • 2 months ago

I tried "sudo xcode-select -s /Library/Developer/CommandLineTools", but whenever I build my project I still get those anoying '_textcoal_nt' warnings (not error with me -> warning: section "__textcoal_nt" is deprecated).

Do I have to rebuild the whole chain after this changes or are my warnings different from the error you are talking about (by the way the warnings' files are some weird "/var/folders/pb/z6r4g8091xg9y9bzl7thgm7r0000gn/T//cczlGTyj.s")?

Reply · Share ›

SolarianProgrammer Mod ➚ Solian • 2 months ago

Thanks. Can you try to compile a C++ file that includes the vector or algorithms headers from the STL ? I've noticed the bug is present only for C++ and only for code that uses the STL.

Reply · Share ›

Solian ➚ SolarianProgrammer • 2 months ago

Yep, Just compiled the tst_lambda.cpp file that you gave in the tutorial, and it seems to have worked. No errors.

Reply · Share ›

SolarianProgrammer Mod ➚ Solian • 2 months ago

Nice. Thanks for letting me know.

Reply · Share ›

Basitha Gajasinghe • 8 days ago

Hi, first of all thank you for your step by step guide. I got the following error. If you could help me I would be much grateful;

../libtool: line 7461: cd: yes/lib: No such file or directory
libtool: error: cannot determine absolute directory name of 'yes/lib'
make[2]: *** [libmpc.la] Error 1
make[1]: *** [all-recursive] Error 1
make: *** [all] Error 2

Reply · Share ›

SolarianProgrammer Mod ➚ Basitha Gajasinghe • 8 days ago

Have you installed the Command Line Tools ?

1 · Reply · Share ›

Travis Chambers ➚ SolarianProgrammer • 7 days ago

I got this same error and I have the Command Line Tools installed. Any other suggestions?

1 · Reply · Share ›

SolarianProgrammer Mod ➚ Travis Chambers • 7 days ago

If nothing works, you can use my already compiled version https://github.com/sol-prog/gc... .

1 · Reply · Share ›

Hardik Suthar ➚ SolarianProgrammer • 3 days ago

I tried to put compiled version "gcc-6.2.0" directory in /usr/local path and also included path in my PATH variable but still its saying that header files are missing.

Mostly it is looking for c header file such as in the include directory, file "cassert" is there but in that its looking for "assert.h" file that is missing.

Please help with this.

Reply · Share ›

SolarianProgrammer Mod ➚ Hardik Suthar • 3 days ago

The compiled version will work only on macOS Sierra and you also need to have previously installed the Command Line Tools as specified in the article.

Reply · Share ›

Hardik Suthar ➚ SolarianProgrammer • 3 days ago

I am trying to copy the content of gcc-6.2.0 directory, but it says the header file not found.

A simple hello world program (cpp) says

fatal error: wchar.h: No such file or directory
#include <wchar.h>
^

Please help.

Reply • Share ›

Johannes P. • a month ago

Thanks A lot! However, I get the "command not found" error still and I completely installed every one of the prerequisites above. Why am I getting command not found still even after waiting an hour to work everything out? BTW, I am using "g++-6.2.0" and says command not found

Reply • Share ›

SolarianProgrammer Mod ➚ Johannes P. • a month ago

You need to add the installation to your path, please read the entire article.

Reply • Share ›

Johannes P. ➚ SolarianProgrammer • a month ago

OOPs i forgot to do the "sudo make install". I am very sorry about this. Thank you for your help, I am deeply sorry about this stupid mistake of mine. It is now working perfectly!

Reply • Share ›

SolarianProgrammer Mod ➚ Johannes P. • a month ago

No problem.

Reply • Share ›

Johannes P. ➚ SolarianProgrammer • a month ago

I did so, in terminal I typed in "export PATH=/usr/local/gcc-6.2.0/bin:$PATH"

Reply • Share ›

Sylvain • a month ago

Hi! Thanks for the post, it's very well explained. I have installed xcode and Command Line Tools, but for unknown reasons, I can't configure gcc ("../configure --prefix=/usr/local/gcc-6.2.0 --enable-cxx").

The last five lines in the terminal indicates the following errors:
"checking whether we are using the GNU C++ compiler... yes
checking whether g++ accepts -g... yes
checking C++ compiler g++ -O2 -pedantic -fomit-frame-pointer -m64 -mtune=sandybridge -march=sandybridge... no
checking C++ compiler g++ -g -O2... no
configure: error: C++ compiler not available, see config.log for details"

My configuration: OSX Sierra 10.12.1 and xcode 8.1. Do you know how to solve this? a problem with the path?

Best regards,
Sylvain

Reply • Share ›

SolarianProgrammer Mod ➚ Sylvain • a month ago

I suppose you have this error when you configure gmp (?), try to use the exact version I've used 6.1.1 .

Reply • Share ›

Sylvain ➚ SolarianProgrammer • a month ago

Sorry, it occurred when I tried to configure gmp. However, I used the version 6.1.1.

Reply • Share ›

SolarianProgrammer Mod ➚ Sylvain • a month ago

Strange. I've compiled gmp 6.1.1 on a few machines without a problem. If you've installed the command line tools

Strange, I've compiled gimp 8.1.1 on a few machines without a problem. If you've installed the command line tools all I can think of is some path conflict on your machine. I tend to use clean installs when a new OS is released.

If you want, I can send you my compiled version.

Reply · Share ›

Sylvain ➚ SolarianProgrammer · a month ago

Yes, it would probably a conflict path. If you can send me your compiled version it would be very helpful. Do you think I have to reinstall xcode before?

Reply · Share ›

SolarianProgrammer Mod ➚ Sylvain · a month ago

I don't think you need to reinstall your Xcode.

I've uploaded my compiled version on GitHub https://github.com/sol-prog/gc... . Hopefully it will work on your machine.

Reply · Share ›

Alan · a month ago

Hi, I accidentally install this gcc on my mac, now how can I uninstall them from my mac?

Reply · Share ›

SolarianProgrammer Mod ➚ Alan · a month ago

How could you "accidentally" install this GCC when there is no installer ? You need to take a few steps in order to be able to even compile this GCC, no way you could "accidentally" install something.

Anyway, assuming you are not trolling and that you've followed my article you can simply erase the folder in which GCC is put:

```
sudo rm -rf /usr/local/gcc-6.2.0
```

Reply · Share ›

Alan ➚ SolarianProgrammer · a month ago

Thank you very much for the help

Reply · Share ›

Alan ➚ SolarianProgrammer · a month ago

I guessed I use the wrong word "accidentally". I am actually new to everything. I tried have some code run in my mac, but encounter an error: "omp.h" file no found. And I googled this error and found out that I need to update gcc to the new version(the GCC version in my mac is apple gcc 4.2.1). Then I googled how to update my GCC version and found your post. And I followed the instruction here and installed gcc 6.2.0. But the error still existed. So I decided to uninstalled it and tried another version(apple gcc 4.9).

Reply · Share ›

SolarianProgrammer Mod ➚ Alan · a month ago

Actually GCC 6.2.0 has omp.h and the procedure presented in my article will install the latest version of OpenMP. If you've followed my article compiling a C++ program that uses OpenMP is as simple as:

```
g++-6.2.0 program_name.cpp -fopenmp
```

or, for C:

```
gcc-6.2.0 program_name.c -fopenmp
```

Reply · Share ›

akash bachu · a month ago

I followed all the steps to update my gcc but if i try to execute this command
"gcc -o app -fopenmp approxpiOMP.c"
i can see the following error

"clang: error: unsupported option '-fopenmp' "

clang: error: unsupported option '-fopenmp'

please help me to sort this.

Reply • Share ›

SolarianProgrammer Mod ➚ akash bachu • a month ago

Try with "gcc-6.2.0" instead of "gcc". When you simply use "gcc", you are actually calling the default "gcc" from your Mac, which is an alias for "clang", which doesn't support OpenMP. This is the correct command for your example:

```
gcc-6.2.0 -o app -fopenmp approxpiOMP.c
```

Reply • Share ›

akash bachu ➚ SolarianProgrammer • a month ago

Thank you
it is working...

Reply • Share ›

Jay • 2 months ago

I get the following linker error, any help ?

/var/folders/qv/n1r_tfn5323c2t1c_0hwyv0c0000gp/T//cc8UVrD5.s:1:11: warning: section "__textcoal_nt" is deprecated
.section __TEXT,__textcoal_nt,coalesced,pure_instructions
^ ~~~~~~~~~~~~~
/var/folders/qv/n1r_tfn5323c2t1c_0hwyv0c0000gp/T//cc8UVrD5.s:1:11: note: change section name to "__text"
.section __TEXT,__textcoal_nt,coalesced,pure_instructions
^ ~~~~~~~~~~~~~
Undefined symbols for architecture x86_64:
"operator new(unsigned long)", referenced from:
insert(int) in ccSsoP82.o
ld: symbol(s) not found for architecture x86_64
collect2: error: ld returned 1 exit status

Reply • Share ›

David Reis ➚ Jay • 2 months ago

Have you found a solution? I have the same problem.....

Reply • Share ›

Jay ➚ David Reis • a month ago

It was silly, i was using gcc instead of g++ for compiling c++ program :) #facepalm

Reply • Share ›

SolarianProgrammer Mod ➚ Jay • 2 months ago

Where did you get this error ? In what stage ? With what code ? What command ?

Reply • Share ›

Nicolò Cogno • 2 months ago

Hi! Thank you for the guide!!!!! What should i do if i get this error?

[15:47:41 build]> ../configure --prefix=/usr/local/gcc-6.2.0 --enable-cxx

checking build system type... ../config.guess: line 127: 17241 Segmentation fault: 11 ($CC_FOR_BUILD ${dummy}0.s $dummy.c -o $dummy) > /dev/null 2>&1

../config.guess: line 127: 17244 Segmentation fault: 11 ($CC_FOR_BUILD ${dummy}0.s $dummy.c -o $dummy) > /dev/null 2>&1

x86_64-apple-darwin16.0.0

checking host system type... x86_64-apple-darwin16.0.0

checking for a BSD-compatible install... /usr/bin/install -c

checking whether build environment is sane... yes

checking for a thread-safe mkdir -p... ../install-sh -c -d

checking for gawk... gawk

---

see more

Reply • Share ›

SolarianProgrammer Mod ➔ Nicolò Cogno • 2 months ago

Seems that you didn't installed the Command Line Tools as suggested in the article. You will need to open a Terminal and write:

```
xcode-select --install
```

than proceed as before.

Reply • Share ›

Nicolò Cogno ➔ SolarianProgrammer • 2 months ago

You're very fast, thanks!! Unfortunately i did it before!

Last login: Thu Sep 29 15:47:00 on ttys000
[15:58:48 ~]> xcode-select --install
xcode-select: error: command line tools are already installed, use "Software Update" to install updates
[15:58:50 ~]>

Reply • Share ›

SolarianProgrammer Mod ➔ Nicolò Cogno • 2 months ago

For some reason the configure utility can't find GCC which is installed by the command line tools (I mean Apple's GCC). I've compiled GCC 6.2 on a clean Sierra machine a few days ago without a problem.

Reply • Share ›

Nicolò Cogno ➔ SolarianProgrammer • 2 months ago

I really don't know how to solve this :(What would you suggest?

Reply • Share ›

SolarianProgrammer Mod ➔ Nicolò Cogno • 2 months ago

Check your email ...

Reply • Share ›

SolarianProgrammer Mod ➔ Nicolò Cogno • 2 months ago

Try to compile something with Apple's gcc. Make a dummy hello world and see if you can compile it. Something like "g++ hello.cpp"

Reply • Share ›

Nicolò Cogno ➔ SolarianProgrammer • 2 months ago

This works

```
#include <iostream>
using namespace std;
int main() {
cout << "Hello world!" << endl;
}
```

I'll try with the thing that you sent me by email

Reply • Share ›

Tiago Martinho • 2 months ago

Very interesting post! And what if I want to use momentarily this version of the compiler? Without using g++-6.2.0 but using g++ or gcc?

Reply • Share ›

SolarianProgrammer Mod ➔ Tiago Martinho • 2 months ago

You can create an alias (this is a temporary change of name, when you will close your Terminal it will revert). For example:

```
alias gcc="gcc-6.2.0"
alias g++="g++-6.2.0"
```

After you will write the above in your Terminal, when you will write g++, you will actually call g++-6.2.0.

Reply · Share ›

Tiago Martinho ➚ SolarianProgrammer · 2 months ago

Allright! Thanks again!

Reply · Share ›

Categories

Books C++ C++11 C++14 C++17 Charts Clang Cling C# CUDA Fortran Fractals GCC iOS JavaScript Jekyll Linux macOS Matplotlib Multithreading NumPy Objective-C OpenCV OpenGL OpenMP OSX Perlin Posix Productivity Python Raspberry Pi Regex Roguelike Scheme SciPy Swift Videos Windows

Recent posts