Speare Code Editor
The Small IDE for C and C++ Development

Version 0.0.1
Update: 4 Mar 2020

Free, Lightweight, Open Source, Extendable Flexibility

Speare (http://www.sevenuc.com/en/speare.html) is an ultra lightweight code editor and a small IDE that has an efficient code navigation and call routines tracing ability, and has integrated debugging environment for C, C++ and Lua. It was originally developed to providing a native scripting language debugging environment that seamlessly integrated with C and C++, and well supports all kinds of game development that using Lua as its scripting language. It not only has very concise user interface but also has a very flexible architecture to easily add a new programming language in it.

For general information of debugging C and C++ projects with Speare code editor, please refer this document:
http://www.sevenuc.com/download/Small_IDE_for_C_and_C++_development.pdf

If some reason force you must install GCC on your macOS to provide an alternative C and C++ compiler instead of Clang comes with Xcode, this document described how to do it step by step in details. The instructions in this document were tested on High Sierra and they should works on Mojave and Catalina.

Compile and install GCC toolchain on macOS

1. The foundation of macOS programming, customising compiler toolchains

a. prepare tarballs for gcc toolchain:

autoconf-2.69.tar.gz
automake-1.15.tar.gz
libtool-2.4.6.tar.gz
pkg-config-0.28.tar

b. Compile autoconf, automake and libtool by default:

```
./configure
make && make install
```

c. Compile pkg-config with:

```
./configure --with-internal-glib
```

d. Add tool chain executables to PATH:

```
echo 'export PATH=/usr/local/bin:$PATH' >> ~/.zshrc
source ~/.zshrc
```

2. Compile GCC 9 on macOS

Building GCC 9 from sources could take some time, in my case it took about two hours on my MacBook Pro. In order to compile GCC from sources you will need a working C++ compiler. In the remaining of this page 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 execute the following command:

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

Now, let's start by creating a working folder:

```
cd ~
mkdir gcc && cd gcc
```

Next, download and extract a stable version of GCC:

```
curl -O https://ftpmirror.gnu.org/gcc/gcc-9.1.0/gcc-9.1.0.tar.xz
curl -O ftp://gcc.gnu.org/pub/gcc/infrastructure/gmp-6.1.0.tar.bz2
curl -O ftp://gcc.gnu.org/pub/gcc/infrastructure/mpfr-3.1.4.tar.bz2
curl -O ftp://gcc.gnu.org/pub/gcc/infrastructure/mpc-1.0.3.tar.gz
curl -O ftp://gcc.gnu.org/pub/gcc/infrastructure/isl-0.18.tar.bz2
```

We will start by compiling and installing the gmp library:

```
tar -xvf gmp-6.1.0.tar.bz2
cd gmp-6.1.0
./configure --prefix=~/gcc --enable-cxx
make
make install
```

We will do the same steps for mpfr now:

```
cd ..
tar -xvf mpfr-3.1.4.tar.bz2
cd mpfr-3.1.4
../configure --prefix=~/gcc --with-gmp=~/gcc
make
make install
```

Now, we are going to build mpc:

```
cd ..
tar -xvf mpc-1.0.3.tar.gz
cd mpc-1.0.3
./configure --prefix=~/gcc \
  --with-gmp=~/gcc \
  --with-mpfr=~/gcc
make
make install
```

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

```
cd ..
tar -xvf isl-0.18.tar.bz2
cd isl-0.18
./configure --prefix=~/gcc --with-gmp-prefix=~/gcc
make
make install
```

After above steps, we are ready to compile GCC now.

```
cd ..
tar -xvf gcc-9.1.0.tar.xz
cd gcc-9.1.0
./configure --prefix=~/gcc \
  --enable-checking=release \
  --with-gmp=~/gcc \
  --with-mpfr=~/gcc \
  --with-mpc=~/gcc \
  --enable-languages=c,c++ \
  --with-isl=~/gcc \
  --program-suffix=-9.1
```

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

option, e.g --enable-languages=c,c++,fortran to add Fortran compiler. And now, execute the commands:

```
make
```

This step may take approximately, depending on your computer configuration, an hour or more, and about 5.82GB of your disk space for the building.

When the compile finished, install the compiled gcc in ~/gcc:

```
make install
```

Now, you can use the new compiled gcc by export the PATH environment variable:

```
export PATH=~/gcc/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, e.g:

```
echo 'export PATH=~/gcc/bin:$PATH' >> ~/.bash_profile
source ~/.bash_profile
```

In your Makefile, use the new compiled gcc-9.1 or g++-9.1 and set the include path:

```
CC = g++-9.1
CXXFLAGS += -std=c++11 -I~/gcc/include/c++/9.1.0
```

After above settings, you can develop C and C++ with a stable version of GCC that compiled by yourself now.