

# VG101 — Introduction to Computer and Programming

## Lab 3

Instructor: Manuel

TA: Zhengyuan — UM-JI (Summer 2019)

### Goals of the lab

- Install C/C++ Compiler
- Install C/C++ IDE

## 1 Foreword

After spending some time in Matlab, Haruka, Kana, and Chiaki decided to challenge themselves with a lower-level language called C.

They spent a huge amount of time searching for something like Matlab for C, but they found that there might be no such counterpart, so they turned to Jane for help.

## 2 Background

Moving from Matlab to C, you might not understand why we need all these complex stuffs instead of a single software like Matlab. To understand this, first look at 2 notable differences between Matlab and C

- Matlab is interpreted, while C is compiled.
- Matlab is a proprietary commercial software Developed by MathWorks. C is standardized by ISO, with multiple compiler implementations available.

Matlab, as a single software, integrates a Graphical User Interface (GUI), an interpreter, and other tools such as linter. The GUI is the place where you write your code and click all the bottoms. When you click the *Run* bottom to run scripts or press *Enter* to execute commands, the interface would invoke the back-end interpreter to run your code and display the result of the interpreter to you. When you are writing code, the interface would invoke the back-end linter to check the syntax of your code and display the result of the linter at the warning bar on the right of the editor.

Then we move to C. You may have noticed that the core component for Matlab development is not the user interface, but the interpreter, which interprets your Matlab code line by line when you run your code. The core component for C development, on the other hand, is the compiler. The compiler compiles your C code into executable assembly code and stores the assembly code on your file system. You would not need a software interpreter to run the assembly code, since the hardware on your system can directly interprets assembly code.

Since C is not a proprietary commercial software, there is not an official implementation of the compiler. Common implementations include GNU Compiler Collection (GCC), clang/LLVM, Intel C Compiler, and Microsoft Visual C++. There is an organization called ISO which sets the standard of the language. However, nobody can force all the implementations to follow the standard. Some would not support some features in the standard, and some would support some features not in the standard. As a result, code that works with one compiler may not work with other compilers, especially when the code uses non-standard features.

The official compiler implementation of this course is GCC. It was initially developed for Linux by GNU,

then was migrated to Mac OS and Windows. It has a relatively good support for the standard. It does have some non-standard features, which however, can be easily disabled by adding options. It can help you to build a good habit of writing standard C/C++ code.

Unlike Matlab, GCC does not have a GUI, but a Command Line Interface (CLI). To use GCC to compile your code, you would need to type command in your terminal. GCC does not provide you with an editor, so you cannot use it to write code like what you do with Matlab. Technically you can use any plain text editor to write code and then use GCC to compile them. There are, however, Integrated Development Environment (IDE) and source code editors that provide special support for code development. An IDE is usually language-specific and out of the box. A source code editor is usually cross-language, cross-platform, lightweight, and configurable. It is worth noting that most of the IDEs and source code editors provide little back-end functionality themselves. CLion, as a heavy IDE, still needs a stand-alone compiler to have the work done. Most of the software act as intermediaries. They accept your inputs and invoke the back-end components accordingly.

## 3 GNU Compiler Collection (GCC)

### 3.1 Installation

When they asked help from Jane, they did not mention what operation system they were using. Considerate as Jane was, she offered instructions of different operation systems.

#### 3.1.1 Windows

Windows users need to download mingw-w64, which is a Windows implementation of GCC. You can download and install mingw-w64 [here](#). Use the configuration shown in Figure 1.

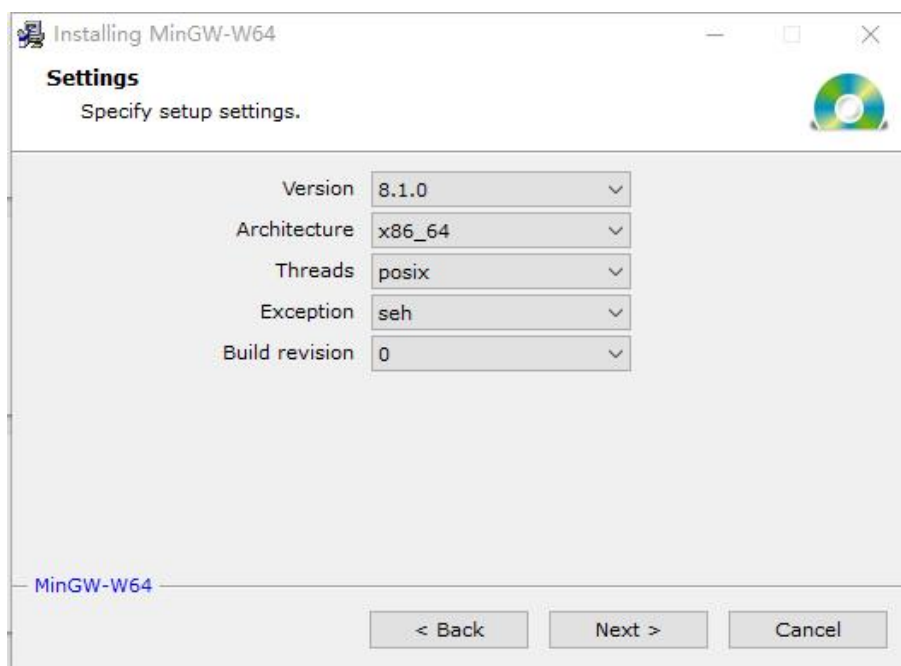


Figure 1: mingw-w64 configuration

**Please pay attention to the destination folder you choose in the installation process.** Since you

may need to find where you have installed it later.

In the installation process, if an error message like **ERROR res** occurs, don't panic. You should go to the destination folder you have chosen to uninstall all the previously downloaded files (Fig 2) and then go through the installation process again from the beginning<sup>1</sup>.

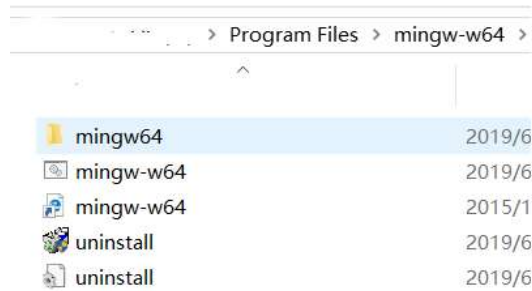


Figure 2: Uninstall if error occurs

### 3.1.2 Mac OS

Mac OS will alert you when you enter a command in the terminal that requires Command Line Developer Tools. For example, you can enter `gcc` in your terminal. If you don't have the tools installed, you will see something like Figure 3.

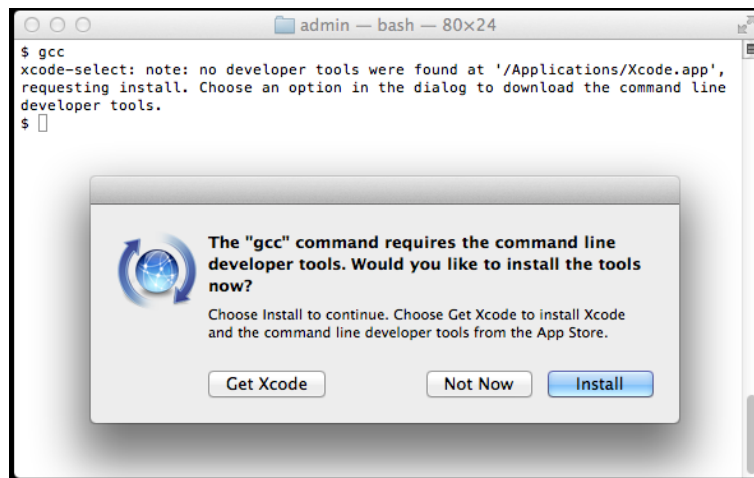
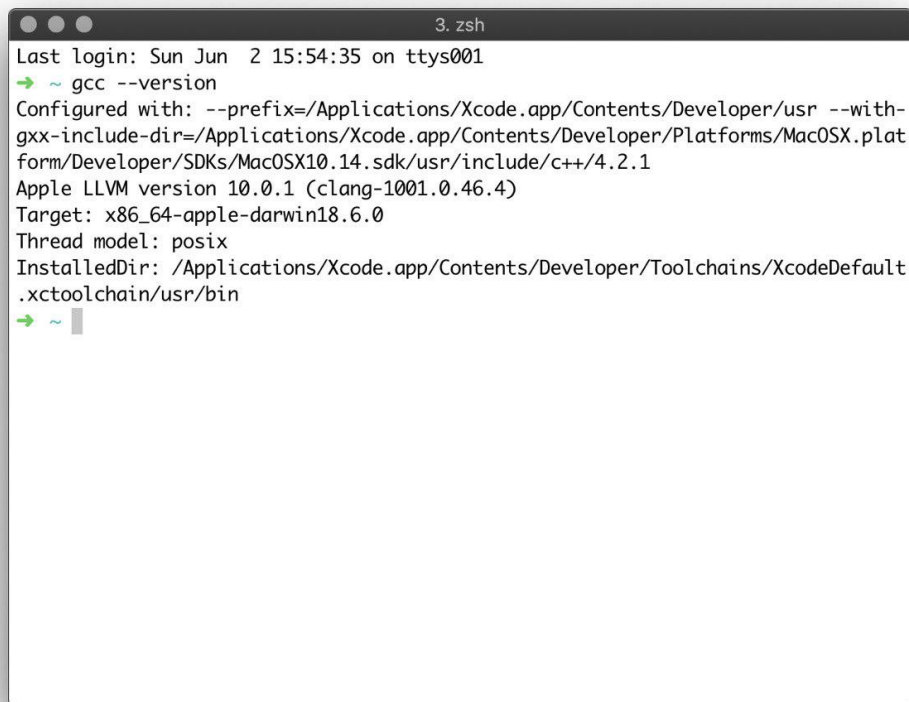


Figure 3: Xcode Command Line Tools

Click "Install" to download and install Command Line Developer Tools.

The instructions in the alert box are confusing. If you click "Get Xcode", besides the Command Line Developer Tools, the Xcode IDE would also be installed, which is not a must. If you would like to use CLion IDE or other source code editors, just click "Install" to install the Command Line Developer Tools. After the tools are installed, if you enter `gcc --version` in your terminal, you can see something like Figure 4 (they may not be strictly the same).

<sup>1</sup>If you care about the reason: When MinGW find something is going wrong it will give an error message and then **skip** it and it is difficult to let it specifically re-handle it.

A terminal window titled '3. zsh' showing the output of the 'gcc --version' command. The output includes the last login time, the gcc version (10.0.1), the target architecture (x86\_64-apple-darwin18.6.0), and the installation path. The prompt is '~' followed by a cursor.

```
3. zsh
Last login: Sun Jun  2 15:54:35 on ttys001
➔ ~ gcc --version
Configured with: --prefix=/Applications/Xcode.app/Contents/Developer/usr --with-
gxx-include-dir=/Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.plat
form/Developer/SDKs/MacOSX10.14.sdk/usr/include/c++/4.2.1
Apple LLVM version 10.0.1 (clang-1001.0.46.4)
Target: x86_64-apple-darwin18.6.0
Thread model: posix
InstalledDir: /Applications/Xcode.app/Contents/Developer/Toolchains/XcodeDefault
.xctoolchain/usr/bin
➔ ~
```

Figure 4: gcc version

### 3.1.3 Linux

GCC is pre-installed on most Linux distributions like Ubuntu. If you are using distributions like Arch Linux, you are unlikely to need help on GCC installation.

## 3.2 Basic Usage

As we mentioned earlier, GCC only has a command line interface. To compile code using GCC, you need to type commands in your terminal. Let us look at some examples.

```
gcc hello_world.c
```

In the command above, you type `gcc` to invoke the compiler. Then you feed the compiler with the file name of the source code. The compiler would read the source code, compile, and dump the executable assembly code to a default file, `a.out` for Linux, `a.exe` for Windows.

To specify the name of the executable, use the option `-o`.

```
gcc hello_world.c -o hello_world.exe
```

In the command above, the output file is no longer `a.out` or `a.exe`, but `hello_world.exe`. Note that if you type something like `gcc hello_world.c -o hello_world.c`, `hello_world.c` would be overwritten by the output of the compilation.

To run the executable, type `./executable_name` for Linux and Mac OS, or `.\executable_name` for Windows.

If you split your code into multiple files,

```
gcc main_body.c function1.c function2.c
```

Note that there should be exactly 1 main function among all the files you provide.

To write code with higher quality, we want the compile to give us more warnings.

```
gcc hello_world.c -Wall
```

And even more,

```
gcc hello_world.c -Wall -Wextra -Werror -pedantic -Wno-unused-result
```

Note that if your code cannot compile at this warning level, your code might be rejected by JOJ.

To specify a C standard

```
gcc hello_world.c -std=c11
```

For a complete list of available options, you can refer to [this site](#)

## 4 From CLI to GUI

Theoretically, GCC is enough for C/C++ development. In practice, you might not be used to a command line interface. If so, you would like to have an IDE or a source code editor as an intermediary between you and the command line interface.

Since IDE might be easier for beginners, we only give detailed instructions for CLion here. Moreover, we are strongly against using Microsoft Visual Studio in this course, because it does not follow the standard tightly, and you may end up finding your code incompatible on different platforms or JOJ if you do not know what you are exactly doing.

### 4.1 CLion

CLion is a cross-platform IDE for C and C++. It is not a free software, but it's free for educational use. You can use your sjtu email to apply for a free license.

#### 4.1.1 Applying for license

The main steps for applying the license is shown below.

1. Visit <http://www.jetbrains.com/student/>.
2. Click **Apply Now**, and then apply with the SJTU email address and your personal information.
3. Receive an email in your SJTU mailbox and open the **Confirm Request** url (Fig 5).

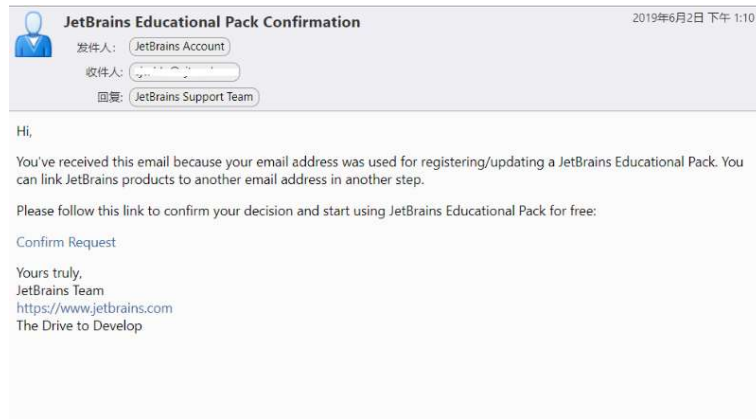


Figure 5: Confirmation email

4. Scroll to the end of the confirmation page and **accept** the policies, and then create an jetbrains account with your sjtu email address and your password
5. Then you will receive the license. You can click the **CLion** button to download it (Fig 6).

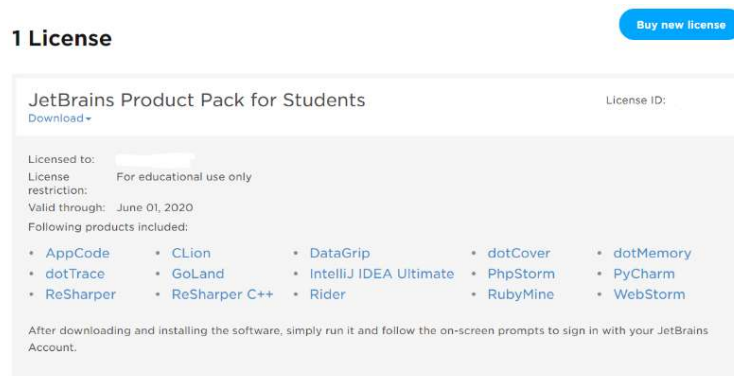


Figure 6: License

Moreover, SJTU Innovation Center <http://lic.sjtu.edu.cn/> offers you a Chinese version of installation guide, you may refer to that if you wish.

#### 4.1.2 Configure you CLion

After the installation process, at the first time you open your CLion, you may be asked to configure several things. You may follow the steps below.

1. It will first ask whether you need to import setting. Please choose do not import settings (Fig 7).

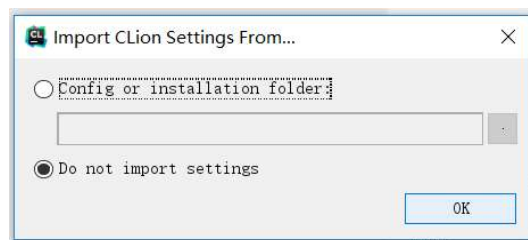


Figure 7: Import setting

2. When you are asked to configure toolchain. Highly likely that CLion can detect previously installed compiler automatically. If it fails to do so, you need to manually choose the directory where your mingw or gcc was installed (Fig 8).

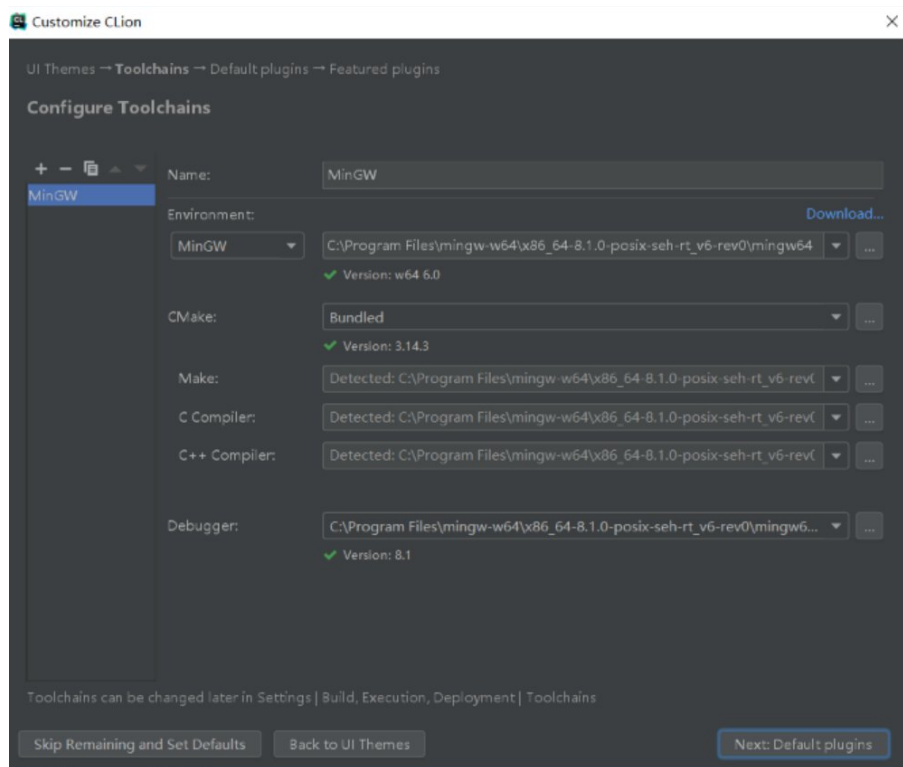


Figure 8: Tool-chains setting

3. At last, you will be asked to activate CLion. In this step, you can simply enter you jetbrains account and password.

#### 4.1.3 Start your first project

After configuring correctly, you may see the following welcome page (Fig 9).

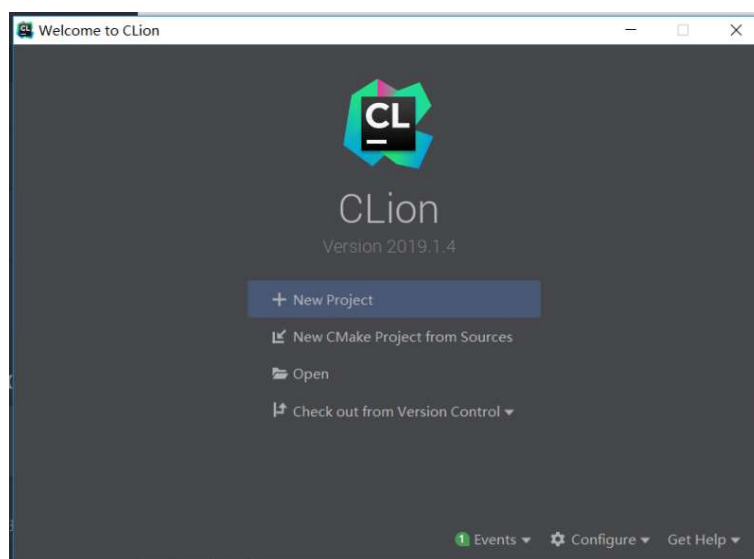


Figure 9: Welcome page

You can click **New Project** to create a new project, and then set the project as follows (Fig 10).

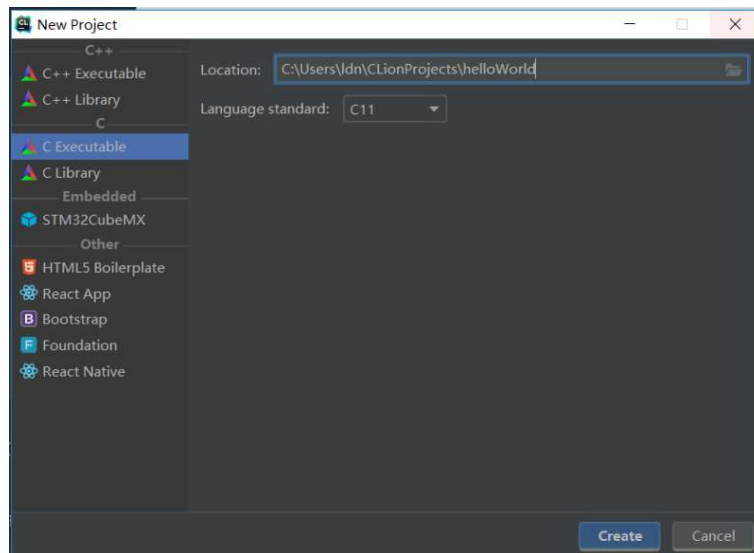


Figure 10: Setting for new project

After finishing everything, you can simply run *main.c* by clicking the green triangle button on the top right corner to check whether the whole installation process is correct or not.

## 4.2 Other Source Code Editors

There are many source code editors available for code development. Some common ones are Atom, Visual Studio Code, Sublime Text, and Vim. Most of them are not out-of-the-box for C development, and need your configuration. To learn these software, you need to be comfortable working with command line, and read their documentation. We do not give detailed instructions here, but some starting points.

- Visual Studio Code: [C/C++ for Visual Studio Code](#)
- Sublime Text 3: [Documentation - Sublime Text](#)
- Atom: [Atom Flight Manual](#)