C++ at the CHACAL School 2024: quickstart guide

This guide contains instructions on how to set up your laptop for coding, compiling and running C++ programs for the CHACAL school. We may have some lab computers available, but if that doesn't happen you will be paired up with someone with similar abilities as you to do the exercises together.

In Visual Studio Code (for all operating systems), make sure that you have the C/C++ extensions installed. You can also install "Intellisense for C++", which is an autocomplete/helper tool that will save you some googling.

Instructions to set up your own laptop

Mac

Installing an editor / Integrated Development Environment (IDE) on a mac

If you have a mac, you can install the Visual Studio Code IDE with a g++ compiler as back-end. You can use visual studio code (https://code.visualstudio.com) as a mac package; an open source IDE that is in very common use with the gcc11 compilers. You can also install the desktop C++ development option as this will give you several helpful tips while writing your code. Visual Studio Code is straightforward to install, but setting up the compilers is slightly more complex: please read on.

Important note: on a mac, it's better to *not* use XCode as a development platform or clang as a compiler, as this leads to different error messages with respect to those using g++.

Installing a compiler on mac

Homebrew installation

Open a terminal window, and copy one of the two following into into a terminal window:

For MacOS Catalina, macOS Mojave, and MacOS Big Sur and MacOS Monterey:

```
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"
```

For macOS High Sierra, Sierra, El Capitan, and earlier:

```
/usr/bin/ruby -e "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

XCode requirements

Then, let the installation program install what you need from xcode.

In a terminal window, type:

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

(we also remind you not to use XCode as a development platform as it generally comes with clang)

Note: it's better to uninstall/reinstall xcode to make sure you have the most recent version. You can do that with:

```
sudo rm -rf /Library/Developer/CommandLineTools
xcode-select --install
```

(this is for users that don't usually do much with CommandLineTools and have no special configuration - take care and do backup anything you'd like to keep or port if you're not in this category!)

gcc/g++ from homebrew

Then, install the gcc/g++ suite from homebrew:

In a terminal window, type:

```
brew install  gcc@11
```

It may take a while for it to complete, but at the end you should have the following file (check in a terminal with "ls" if you do have it, and it doesn't return that it doesn't find the file):

```
/usr/local/bin/g++-11
```

In more recent version of homebrew, you may find this file in a different brew directory, e.g. /usr/local/Cellar/ or /opt/homebrew/bin/ - we can sort this out during the course.

Note: Another option to install the compiler is to use MacPorts (ports.macports.org/port/gcc11) but we haven't tested that this works yet.

Using a compiler with automated compilation in VS Code (optional)

Finally, you can tell Visual Studio Code to use this compiler, if you're using that as an Integrated Development Environment (IDE).

You should then open VS Code and go to Settings, which is the wheel on the bottom of the window (this was found as File->Preferences->Settings in older versions), The main window of VS Code will turn into a list of settings.

In the top bar, search for "compiler" and you should see C_Cpp › Default: Compiler Path. Put in:

```
[your installation path from brew]/g++-13
```

Then write a simple code in the editor window, e.g. the one below:

```
// L1/simple.cpp
// Another particularly simple example of C++ in action!
// Niels Walet, last updated 22/01/2023
#include <iostream>

int main()
{
const int current_year = 2023;
std::cout << "C++ is the best programming language in "<<current_year<<"!"<<
std::endl;
return 0;
}
```

Then, try to compile it (this is called a "build task" in Visual Studio).

What a build task does is calling the compiler with some flags (you could also do the same in the terminal window yourself).
By default, you will have clang installed or mac's version of g++ (which is basically clang so it's not good) and Visual Studio will use that.

But if your installation of gcc/g++11 or gcc12 went well, you should see g++11 or 12 in the drop-down menu. You should also be able to choose g++11 as the option if you are starting Visual Studio for the first time.

Windows

Installing a compiler

We suggest to install the latest gcc compilers using mingw. Simple instructions on how to do this are [at this link]

(https://code.visualstudio.com/docs/cpp/config-mingw#_installing-the-mingww64-toolchain

Pay particular attention to how you set up PATH variables because this will allow your Windows installation to "see" the compilers from everywhere. You will need administrator privileges for this.

If you have a Windows computer where you don't have administrator privileges, we will pair you up with someone else with similar abilities as you so that you can work with them.

Installing a text editor / IDE

We suggest you use an Integrated Development Environment (IDE) like Visual Studio Code. You don't need to use all the features (e.g. automatic compilation etc...) and we will not necessarily do so in the course, but this is convenient as it:

- provides a terminal as well where you can enter your commands
- automatically build (compile) your code and run it
- have a visual interface for a debugger

You can download and install Visual Studio Code from here.

This website also has a tutorial on how to set up and configure the IDE to work with the gcc compiler above - if you follow them.

Linux

The installation of the compiler and text editor of your choice will depend on the Linux distribution you have, and it's generally straightforward to do so through the linux package manager of your choice. You will need to install gcc-11 (a newer series like gcc-13 will also work).

The suggestion if you want an editor is to use Visual Studio Code, instructions to install and set up are here.

If you don't want to use Visual Studio Code, vim is better than emacs. Fight me.

Instructions for lxplus (for users with a CERN account and access to the CERN computing cluster)

If you have at any point in time worked at CERN and still have a computing (lxplus) account, all you need to do is to set up a recent compiler, and you should be all set to start! This is generally done for you in the most recent lxplus nodes (lxplus9) but you can also choose your the compiler from one of the most recent LCG releases.

An introduction to the LCG releases and their use in practice is here

In practice, all you need to do is this:

```
source /cvmfs/sft.cern.ch/lcg/contrib/gcc/13/x86_64-el9.sh
```

Checking that the exercises work

Installing git / getting the exercises

You will need to download (clone) the repository where the exercises are via Git, which also will need to be installed on your laptop. You should already have it if you followed the "mac" instructions above, for everything else you can find instructions here.

Once Git is available, you can write this in a terminal, make sure first to go to the top directory where you want the exercises to be in (you can call this directory CHACAL and keep all your further tutorials in the same place):

```
git clone https://github.com/hsf-training/cpluspluscourse.git
cd cpluspluscourse/exercises
```

If for some reason you can't install Git, you can download the tarball with the exercises from this link -> "Code" button -> Download ZIP.

Testing that your installation / exercises will work

This part is taken from the exercises README

- **Please run `check_setup.sh` to check your setup on Linux / Mac / Windows.**
  - In Windows, it's best to do this in the terminal of Visual Studio Code.
  - Note that the optional tools are not required for the essentials course, so it's fine if they are not there.
- if that works, go to the directory `exercises/hello`
- follow the `README.md`

How to compile and run?

```
cd exercises/hello
make
export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:.
./hello
```

Optionally: check for valgrind/cppcheck

- `valgrind --tool=callgrind ./hello; kcachegrind`
- `cppcheck .`

Installing Git

Instructions on how to install Git on Linux, Mac and Windows are here: https://git-scm.com/book/en/v2/Getting-Started-Installing-Git.

Troubleshooting

A reminder: make it easy for others to help you

Make sure that your question is posed as a Minimal Reproducible Example:
https://stackoverflow.com/help/minimal-reproducible-example

VS Code can't find the compilers in the drop down menu

If a .cpp file is opened in standalone mode without opening the corresponding folder, VS Code won't allow build tasks to be ran and it looks like the compilers aren't picked up (probably because of a security popup where one needs to press OK when opening a folder). The solution is to open a folder from the File menu and put your .cpp file in there.

Conda can be a slithering problem

If for any Python courses you've installed Conda and have an environment set up for you at start up, it may mess with your environment variables - some errors we've seen include mention of an *unsupported tapi file type*. You should edit your .bash_rc or .zsh_rc file in a terminal (which tells your system what to load at startup), it's usually in your home/ directory. The easiest way to go ahead is to move the ./bash_profile file to ./bash_profile_conda, and create a new empty ./bash_profile, and then restart VS Code.

Mac wchar.h issues

If you have an error like this on the hello world program:

```
/usr/local/Cellar/gcc/13.1.0/bin/g++-13 --std=c++20 -c -g -Wall -Wextra -o
hello.o hello.cpp
```

```
In file included from
/usr/local/Cellar/gcc/13.1.0/include/c++/13/bits/postypes.h:40,
                 from /usr/local/Cellar/gcc/13.1.0/include/c++/13/iosfwd:42,
                 from /usr/local/Cellar/gcc/13.1.0/include/c++/13/ios:40,
                 from /usr/local/Cellar/gcc/13.1.0/include/c++/13/ostream:40,
                 from /usr/local/Cellar/gcc/13.1.0/include/c++/13/iostream:41,
                 from hello.cpp:3:
/usr/local/Cellar/gcc/13.1.0/include/c++/13/cwchar:44:10: fatal error: wchar.h:
No such file or directory
   44 | #include <wchar.h>
      |          ^~~~~~~~
compilation terminated.
```

it means that your Command Line Tools are too old, and you need to uninstall/reinstall as above. This will usually take 10-15' on a recent mac, and you can find instructions here.