Workshop 1. How to build a simple "Hello world" program in C++

S. Shershakov

October 28, 2019

Ed. 1.0 as of 28.10.2019

1 Prerequisites, Goals, and Outcomes

Prerequisites Students should have mastered the following prerequisite skills: basic programming in Python.

Goals This assignment is designed to reinforce the student's understanding of the process of building C++ programs with CMake meta-building system.

Outcomes Students successfully completing this assignment would master the following: understanding how to build C++ programs.

2 Description

In order to start developing C++ applications a set of tools have to be installed and properly configured. They are:

- *toolchain* is a C++-compiler and related tools needed to preprocess, compile, and link your program code to an executable application;
- *a building system* is an auxiliary tool that helps to manage complex projects that consist of multiple units and targets;
- *IDE* is integrated development environment, a complex tool that provides functionality of code editing, invoking a *building system* together with a *toolchain*, program debbugging, integration with a version-control system[1] and so on.

[1] Such as GIT, available at `https://git-scm.com/`

In this course we will use the GCC toolchain and `g++` compiler as primary tools. The toolchain is available as:

- *gcc* package in *nix-base systems; check the availability of invoking the command: `g++ -version`, in the case of success it outputs the actual version of your C++ compiler; the version must be equal or greater than *7.3.0*;
- external application for Windows systems called MinGW; download the latest 32-bit version[2] and setup it according to the installation instructions;
- in Apple systems, GCC there is commonly substituted with Clang toolchain, which is compatible with GCC and also OK for the course purposes[3].

[2] http://www.mingw.org/

[3] Consult with a trainer in case of any problems.

The CMake system is quite often pre-installed in *nix- and Apple-systems. If not, see the official documentation on how to install the corresponding package in your system. Windows users are to download[4] CMake tool and install it as a separate application. Both 32- and 64-bit versions are ok regardless of the bit depth of the GCC toolchain.

[4] `https://cmake.org/`

We recommend to use QtCreator[5] as a full-fledge free IDE for mastering C++ projects. You can use any other IDE that supports CMake and GCC, such as JetBrains CLion, but using it remains at your own responsibility.

[5] Community edition is available independently of Qt framework at `https://www.qt.io/download` free of charge.

We recommend to have a sophisticated notepad application such as Notepad++[6] or Sublime Text[7] for general purpose editing.

[6] `https://notepad-plus-plus.org/`

[7] `https://www.sublimetext.com/3`

Finally, you will need a tool to extract zip archives and, for further use, the installed and configured git scm system[8].

[8] Instructions for this will be provided further.

3 *Files*

All files needed to master this task are available as a zip archive named `workshop01.zip` or can be cloned from an associated github-repository[9]. The structure of a project is described in embedded `.md` documents[10].

[9] `https://github.com/pumpkin-code/dsba-wshp01.git`

[10] See `README.md` in the root for the begining.

4 *Tasks*

1. Study the structure of the given project.
2. Study the content of `/helloworld/src/CMakeLists.txt` file. This is an entry point for building your application. Discuss the content with your trainer.
3. Open the file `/helloworld/src/CMakeLists.txt` in your preffered IDE *as a project*. Beware that you haven't opened it just as a simple file. Consult with your trainer if needed.
4. Configure the project: set up the output directory as `/helloworld/build-debug/`.
5. Your first application called "Hello world" is ready for building. Do it by using "Build" command of your IDE. Do not *run* or *run for debug* the application *in order to* build it. *Bulding* and *running* are separated processes and you have to distinguish between them.
6. Figure out whether building the app is done succesfully[11]. If so, run it and observe the output. It must be "Hello world".
7. Modify the code of the application according to the instructions of the trainer. Build and run the app iteratively.
8. Prepare all necessary files for the second target of your project. The files are: `main.cpp`[12], `foo.h`[13], `foo.h`[14] They must be put in `/helloworld/src/ex2/` folder.
9. In `/helloworld/src/CMakeLists.txt` file, uncomment the command `add_executable(ex2...)` in order to activate the second target of your project. Depending on your IDE, you have to reload the CMake file or even to close-and-reopen the `CMakeLists.txt` again. Make sure that your IDE reflects the changes in the file. Make sure that you reflect the changes in the configuration as well :)
10. Practice with debugging essentials according to the instructions of your trainer.

[11] If you have any troubles with building an application, consult the trainer.

[12] A startup *module*.

[13] A *header* with a *prototype/declaration* of another function `int foo();`.

[14] A *module* with an *implementation/definition* of the function `foo()`.