C and C++

Page maintainer: Johan Hidding @jhidding

C++ is one of the hardest languages to learn. Entering a project where C++ coding is needed should not be taken lightly. This guide focusses on tools and documentation for use of C++ in an open-source environment.

Standards

The latest ratified standard of C++ is C++17. The first standardised version of C++ is from 1998. The next version of C++ is scheduled for 2020. With these updates (especially the 2011 one) the preferred style of C++ changed drastically. As a result, a program written in 1998 looks very different from one from 2018, but it still compiles. There are many videos on Youtube describing some of these changes and how they can be used to make your code look better (i.e. more maintainable). This goes with a warning: Don't try to be too smart; other people still have to understand your code.

Practical use

Compilers

There are two main-stream open-source C++ compilers.

- GCC
- LLVM - CLANG

Overall, these compilers are more or less similar in terms of features, language support, compile times and (perhaps most importantly) performance of the generated binaries. The generated binary performance does differ for specific algorithms. See for instance this Phoronix benchmark for a comparison of GCC 9 and Clang 7/8.

MacOS (XCode) has a custom branch of `clang`, which misses some features like OpenMP support, and its own libcxx, which misses some standard library things like the very useful `std::filesystem` module. It is nevertheless recommended to use it as much as possible to maintain binary compatibility with the rest of macOS.

If you need every last erg of performance, some cluster environments have the Intel compiler installed.