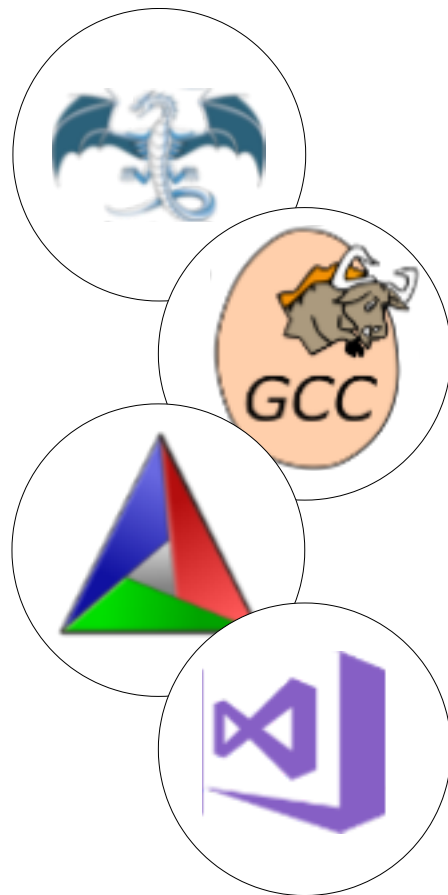


Modern C++ Toolchain

Modern C++ Toolchain

CUHKSZ-HPC Tech sharing

Mar. 24th, 2023



Compiler

- g++: Part of GCC, Licensed as GPL and LGPL , support more architectures than others, linux default, treated as std implementation.
- clang++: Part of LLVM, Licensed as Apache 2.0, modularized, support static analysis and code formatting, support cuda programming.
- MSVC: Microsoft, free but proprietary, hard to download, must-have for Windows programming (C++ programs using Windows API).

Compiler

- Apple Clang: Part of LLVM, but very Apple, not recommended, use the original version.
- EDG eccp: Buy!
- Intel® oneAPI DPC++/C++: Open source, optimized for Intel CPU and GPU. Good in HPC. LLVM as it's backend. Does not support AMD.
- IBM XLC++: Buy!
- Sun/Oracle C++: Free but proprietary.
- Embarcadero C++ Builder: Buy!
- Cray: Can't find download link. Designed for Cray System, LLVM as it's backend.
- Nvidia HPC C++: GPU programming.
- Nvidia nvcc: GPU programming.

Compiler Choice

In most cases, I recommend you to use clang (more friendly error messages)

```
#include <iostream>
int main() {
    int value = 4;
    const char *s = "Hello, World!" + value;
    std::cout << s << std::endl;
    return 0;
}
```

```
> g++ -Wall -Wextra -Wpedantic warning.cpp # gcc version 12.2.1 20230201
> clang++ -Wall -Wextra -Wpedantic warning.cpp # clang version 15.0.7
warning.cpp:4:35: warning: adding 'int' to a string does not append to the string [-Wstring-plus-int]
    const char *s = "Hello, World!" + value;
                                ~~~~~^~~~~~
warning.cpp:4:35: note: use array indexing to silence this warning
    const char *s = "Hello, World!" + value;
                                ^
                                &      [      ]
1 warning generated.
```

<https://easyaspi314.github.io/gcc-vs-clang.html>

Compiler Choice

In most cases, I recommend you to use clang

Easy to download

It is supported by Linux, Windows and MacOS.

What about g++?

Windows needs mingw. mingw is no longer maintained. You will need to use mingw-w64. This software cannot be downloaded alone, and you need to use it with cygwin and msys2.

Other reasons

As of 2023/4/4, the binaries compilation speed by the two have their own advantages and disadvantages.

<https://benchmarksgame-team.pages.debian.net/benchmarksgame/fastest/c.html>

Build System

- GNU Make: Use the familiar Makefile to build C++ projects.

Pros: Everyone is using it, many old projects such as Linux, Sqlite are still using it.

Cons: Slow, not user friendly. You can easily get into trouble when writing Makefile. Such as indentation: you need to use tab instead of space.

- Ninja: A new build system, much faster than GNU Make.

Pros: Fast, easy to use. Does not mess up your terminal.

Cons: Does not support hand writing and string manipulation. You need to use meta build system to generate Ninja build file.

For projects using modern meta build system, I recommend using Ninja as the build system for faster generation.












Meta Build System

- Autotools: An old meta build system, can generate GNU Makefile.
- CMake: A cross-platform meta build system, can generate GNU Makefile or Ninja build file.
- Bazel: Produced by Google, can generate GNU Makefile or Ninja build file.
- Meson: A cross-platform meta build system, can generate GNU Makefile or Ninja build file.
- GN: Produced by Google, can generate Ninja build file.

Autotools

We won't talk about it too much, because it is too old. But it is still used by many projects. So we need to know how to use it.

Such as the HPC cluster management software Slurm.

 Makefile.am	Add make clean-contrib
 Makefile.in	Add auxdir/gtk-2.0.m4 so we can bring it up to date.
 NEWS	Fix typo for initialized in description for ESLURM_PLUGIN_NOT_LOADED.
 README.rst	Docs - Remove DejaGNU from documentation
 RELEASE_NOTES	Fix typo for initialized in description for ESLURM_PLUGIN_NOT_LOADED.
 aclocal.m4	Add auxdir/gtk-2.0.m4 so we can bring it up to date.
 config.h.in	switch/hpe_slingshot - use new cxil_get_svc_rsrc_list routines
 configure	Merge cons_common into cons_tres.
 configure.ac	Merge cons_common into cons_tres.
 make_ref.include	Fix Makefile *.bino generation error for FreeBSD.
 slurm.spec	Merge branch 'slurm-23.02'

Take the PKGBUILD of ArchLinux package as an example, we can see its build process:

```
$ autoreconf

$ ./configure \
  --disable-developer \
  --disable-debug \
  --without-rpath \
  --enable-optimizations \
  --prefix=/usr \
  --sbindir=/usr/bin \
  --sysconfdir=/etc/slurm-llnl \
  --localstatedir=/var \
  --enable-pam \
  --with-hdf5 \
  --with-hwloc \
  --with-rrdtool \
  --with-munge

$ make -j
```

Using ``autoreconf`` is to prevent developers from forgetting to regenerate the ``configure`` script after modification,

or the ``configure`` script is not in the repository at all.

CMake

The de facto standard meta build system for C++.

```
$ cmake \  
-DCMAKE_EXPORT_COMPILE_COMMANDS:BOOL=TRUE \  
-DCMAKE_BUILD_TYPE:STRING=Debug \  
-DCMAKE_C_COMPILER:FILEPATH=/usr/bin/clang \  
-DCMAKE_CXX_COMPILER:FILEPATH=/usr/bin/clang++ \  
-DCMAKE_CXX_FLAGS:STRING="-Wall -Wextra -Wpedantic" \  
-DCMAKE_LINKER:FILEPATH=/usr/bin/ld.lld \  
-Bbuild \  
-G Ninja \  
.  
  
$ cd build  
  
$ ninja
```

Useful Tools

- clangd: LSP server developed based on clang, can provide syntax highlighting, auto completion, jump and other functions.

Why you shouldn't use the cpp intellisense under Microsoft's vscode-cpptools?

```
public int strength;  
public int intelligence;  
public int |outtelligence;
```

Clangd is better in terms of functionality, speed and stability.

If you want to use AI technology to complete the code, my suggestion is to use Github Copilot instead of vscode-cpptools.

clangd uses clang's error reporting method, the error message is more complete and friendly, and the warning can be seen without compilation, saving a lot of time.

Useful Tools

- CodeLLDB: An extension developed based on LLDB, providing debugging functions.

Compared with vscode-cpptools, I have experienced that there is no big difference except for the inconsistent syntax.

In particular, because lldb-mi has been deprecated, if you want to use lldb for debugging, for convenience, I only recommend using CodeLLDB.

GDB is okay for both extensions.

- CMake && CMake Tools

Provide some CMake syntax completion and a graphical interface for quick operations, which is quite convenient, but not necessary.

Useful Tools

- Catch2

A test framework that is easier to use than Google Test. For example, creating a window and destroying a window only needs to be written once and can be applied to all tests.

- Perf

A performance analysis tool on Linux, which can be used to analyze the performance bottleneck of the program.

Generally used in conjunction with cargo-flamegraph to generate flame graphs.

Linux users can try <https://github.com/KDAB/hotspot>. It is graphical and powerful.

- cargo-flamegraph

A rust implementation of flamegraph, with full usability improvements.

<https://github.com/flamegraph-rs/flamegraph>

