Introduction to modern CMake

Oxford Research Software Engineering

Fergus Cooper ~ Graham Lee ~ Thibault Lestang

Problem Statement

You want your C++ code to compile on other computers, not just your laptop.

- group workstation
- HPC compile node
- collaborator laptops

Everyone should end up with a program that behaves the same way, wherever they build.

Enter CMake

You describe *targets* (what to build), *inputs* (the sources files), and *configuration* (what libraries to use, what compiler settings, etc.).

CMake uses that with its own *rules* (how to turn sources into programs) to generate makefiles, IDE projects, or other outputs. CMake doesn't build your project itself!

CMake works on Linux, Windows, macOS and more.

Getting Started

Checkpoint 0 is a simple “hello, world” program written in C++. Let’s use CMake to build it.

```
$ cd checkpoint_0
```

`CMakeLists.txt`

This is where you write the definitions of your targets and configuration.

Let's look at the sample `CMakeLists.txt` line by line.

CMake has changed a lot

```
cmake_minimum_required(VERSION 3.17)
```

Tells CMake which version we used, affecting the features available and the interpretation of `CMakeLists.txt`

Define a project

```
project(IntroCMakeCourse LANGUAGES CXX)
```

We have a project called `IntroCMakeCourse`, in the C++ language.

Configure the compiler

```
set(CMAKE_CXX_STANDARD 17)
```

We're using the C++17 language dialect.

Tell it what to build

```
add_executable(main_executable main.cpp)
```

There is a program, called `main_executable`, which depends on the source code in `main.cpp`

Using CMake

It's typical to build "out of tree", by running CMake in a separate place. Keeps generated files out of your source folder.

```
checkpoint0$ mkdir build
checkpoint0$ cd build
build$ cmake ..
[...]
-- Build files have been written to: /Users/gralee/OxfordRSE/IntroCMakeCou
```

Build your project

CMake only generated the build script, it didn't actually compile anything.

```
build$ make
[...]
[100%] Built target main_executable
build$ ./main_executable
Checkpoint 0
Hello, World!
```

Choosing a generator

CMake can create more than Makefiles. It can generate IDE projects, or build descriptions for the fast Ninja tool.

```
build$ cmake -G Ninja ..
[...]

build$ ninja
[2/2] Linking CXX executable main_executable
```

Setting configuration

You (and users) can override choices made by CMake using the `-D` argument.

```
build$ cmake -DCMAKE_CXX_COMPILER=/usr/local/bin/g++-10 ..
-- Configuring done
You have changed variables that require your cache to be deleted.
Configure will be re-run and you may have to reset some variables.
The following variables have changed:
CMAKE_CXX_COMPILER= /usr/local/bin/g++-10

-- The CXX compiler identification is GNU 10.2.0
[...]
```

That's all, folks

This was only the tiniest tip of the modern CMake iceberg. There are so many great resources available, and here are just a few of them:

- The CMake documentation (link)
- Professional CMake: A Practical Guide (link)

Thank you for coming!