Hello CMake

Table of Contents

Introduction

Shows a very basic hello world example.

The files in this tutorial are below:

```
A-hello-cmake$ tree
.
├── CMakeLists.txt
├── main.cpp
```

- CMakeLists.txt - Contains the CMake commands you wish to run
- main.cpp - A simple "Hello World" cpp file.

Concepts

CMakeLists.txt

CMakeLists.txt is the file which should store all your CMake commands. When cmake is run in a folder it will look for this file and if it does not exist cmake will exit with an error.

Minimum CMake version

When creating a project using CMake, you can specify the minimum version of CMake that is supported.

```
cmake_minimum_required(VERSION 3.5)
```

Projects

A CMake build can include a project name to make referencing certain variables easier when using multiple projects.

```
project (hello_cmake)
```

Creating an Executable

The add_executable() command specifies that an executable should be build from the specified source files, in this example main.cpp. The first argument to the add_executable() function is the name of the executable to be built, and the second argument is the list of source files to compile.

```
add_executable(hello_cmake main.cpp)
```

NOTE

A shorthand that some people use is to have the project name and executable name the same. This allows you to specify the CMakeLists.txt as follows,

```
cmake_minimum_required(VERSION 2.6)
project (hello_cmake)
add_executable(${PROJECT_NAME} main.cpp)
```

In this example, the project() function, will create a variable ${PROJECT_NAME} with the value hello_cmake. This can then be passed to the add_executable() function to output a 'hello_cmake' executable.

Binary Directory

The root or top level folder that you run the cmake command from is known as your CMAKE_BINARY_DIR and is the root folder for all your binary files. CMake supports building and generating your binary files both in-place and also out-of-source.

In-Place Build

In-place builds generate all temporary build files in the same directory structure as the source code. This means that all Makefiles and object files are interspersed with your normal code. To create an in-place build target run the cmake command in your root directory. For example:

```
A-hello-cmake$ cmake .
-- The C compiler identification is GNU 4.8.4
-- The CXX compiler identification is GNU 4.8.4
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Configuring done
-- Generating done
-- Build files have been written to: /home/matrim/workspace/cmake-examples/01-basic/A-
hello-cmake

A-hello-cmake$ tree
.
├── CMakeCache.txt
├── CMakeFiles
│   ├── 2.8.12.2
│   │   ├── CMakeCCompiler.cmake
│   │   ├── CMakeCXXCompiler.cmake
│   │   ├── CMakeDetermineCompilerABI_C.bin
│   │   ├── CMakeDetermineCompilerABI_CXX.bin
│   │   ├── CMakeSystem.cmake
│   │   ├── CompilerIdC
│   │   │   ├── a.out
│   │   │   └── CMakeCCompilerId.c
│   │   └── CompilerIdCXX
│   │       ├── a.out
│   │       └── CMakeCXXCompilerId.cpp
│   ├── cmake.check_cache
│   ├── CMakeDirectoryInformation.cmake
│   ├── CMakeOutput.log
│   ├── CMakeTmp
│   ├── hello_cmake.dir
│   │   ├── build.make
│   │   ├── cmake_clean.cmake
│   │   ├── DependInfo.cmake
│   │   ├── depend.make
│   │   ├── flags.make
│   │   ├── link.txt
│   │   └── progress.make
│   ├── Makefile2
│   ├── Makefile.cmake
│   ├── progress.marks
│   └── TargetDirectories.txt
├── cmake_install.cmake
├── CMakeLists.txt
├── main.cpp
├── Makefile
```

Out-of-Source Build

Out-of-source builds allow you to create a single build folder that can be anywhere on your file system. All temporary build and object files are located in this directory keeping your source tree clean. To create an out-of-source build run the cmake command in the build folder and point it to the directory with your root CMakeLists.txt file. Using out-of-source builds if you want to recreate your cmake environment from scratch, you only need to delete your build directory and then rerun cmake.

For example:

```
A-hello-cmake$ mkdir build

A-hello-cmake$ cd build/

matrim@freyr:~/workspace/cmake-examples/01-basic/A-hello-cmake/build$ cmake ..
-- The C compiler identification is GNU 4.8.4
-- The CXX compiler identification is GNU 4.8.4
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Configuring done
-- Generating done
-- Build files have been written to: /home/matrim/workspace/cmake-examples/01-basic/A-
hello-cmake/build

A-hello-cmake/build$ cd ..

A-hello-cmake$ tree
.
├── build
│   ├── CMakeCache.txt
│   ├── CMakeFiles
│   │   ├── 2.8.12.2
│   │   │   ├── CMakeCCompiler.cmake
│   │   │   ├── CMakeCXXCompiler.cmake
│   │   │   ├── CMakeDetermineCompilerABI_C.bin
│   │   │   ├── CMakeDetermineCompilerABI_CXX.bin
│   │   │   ├── CMakeSystem.cmake
│   │   │   ├── CompilerIdC
│   │   │   │   ├── a.out
│   │   │   │   └── CMakeCCompilerId.c
│   │   │   └── CompilerIdCXX
│   │   │       ├── a.out
│   │   │       └── CMakeCXXCompilerId.cpp
│     │     ├──── cmake.check_cache
│     │     ├──── CMakeDirectoryInformation.cmake
│     │     ├──── CMakeOutput.log
│     │     ├──── CMakeTmp
│     │     ├──── hello_cmake.dir
│     │     │     ├──── build.make
│     │     │     ├──── cmake_clean.cmake
│     │     │     ├──── DependInfo.cmake
│     │     │     ├──── depend.make
│     │     │     ├──── flags.make
│     │     │     ├──── link.txt
│     │     │     └──── progress.make
│     │     ├──── Makefile2
│     │     ├──── Makefile.cmake
│     │     ├──── progress.marks
│     │     └──── TargetDirectories.txt
│     ├──── cmake_install.cmake
│     └──── Makefile
├──── CMakeLists.txt
├──── main.cpp
```

All examples in this tutorial will use out-of-source builds.

Building the Examples

Below is sample output from building this example.

```
$ mkdir build

$ cd build

$ cmake ..
-- The C compiler identification is GNU 4.8.4
-- The CXX compiler identification is GNU 4.8.4
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Configuring done
-- Generating done
-- Build files have been written to: /workspace/cmake-examples/01-
basic/hello_cmake/build

$ make
```

```
Scanning dependencies of target hello_cmake
[100%] Building CXX object CMakeFiles/hello_cmake.dir/hello_cmake.cpp.o
Linking CXX executable hello_cmake
[100%] Built target hello_cmake

$ ./hello_cmake
Hello CMake!
```