ttroy50 / **cmake-examples**

Branch: **master** **cmake-examples** / 01-basic / **J-building-with-ninja** /

Create new file **Find file** **History**

ttroy50 update some examples to require cmake v3 Latest commit `9001357` on 17 Aug 2016

..

CMakeLists.txt	update some examples to require cmake v3	a year ago
README.adoc	link files from intro	2 years ago
main.cpp	example for using clang and ninja	2 years ago
pre_test.sh	added docker files to allow testing with different cmake versions for #8	a year ago
run_test.sh	check if ninja supported by cmake	2 years ago

README.adoc

Building with ninja

Table of Contents

Introduction

As mentioned, CMake is a meta-build system that can be used to create the build files for many other build tools. This example shows how to have CMake use the ninja build tool.

The files in this tutorial are below:

```
$ tree
.
├── CMakeLists.txt
├── main.cpp
```

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

Concepts

Generators

CMake generators are responsible for writing the input files (e.g. Makefiles) for the underlying build system. Running `cmake --help` will show the generators available. For cmake v2.8.12.2 the generators supported on my system include:

```
Generators

The following generators are available on this platform:
  Unix Makefiles              = Generates standard UNIX makefiles.
  Ninja                       = Generates build.ninja files (experimental).
  CodeBlocks - Ninja          = Generates CodeBlocks project files.
  CodeBlocks - Unix Makefiles = Generates CodeBlocks project files.
  Eclipse CDT4 - Ninja         = Generates Eclipse CDT 4.0 project files.
  Eclipse CDT4 - Unix Makefiles
                               = Generates Eclipse CDT 4.0 project files.
  KDevelop3                    = Generates KDevelop 3 project files.
  KDevelop3 - Unix Makefiles   = Generates KDevelop 3 project files.
  Sublime Text 2 - Ninja       = Generates Sublime Text 2 project files.
  Sublime Text 2 - Unix Makefiles
                               = Generates Sublime Text 2 project files.Generators
```

As specified in this post, CMake includes different types of generators such as Command-Line, IDE, and Extra generators.

Command-Line Build Tool Generators

These generators are for command-line build tools, like Make and Ninja. The chosen tool chain must be configured prior to generating the build system with CMake.

The supported generators include:

- Borland Makefiles
- MSYS Makefiles
- MinGW Makefiles
- NMake Makefiles
- NMake Makefiles JOM
- Ninja
- Unix Makefiles
- Watcom WMake

IDE Build Tool Generators

These generators are for Integrated Development Environments that include their own compiler. Examples are Visual Studio and Xcode which include a compiler natively.

The supported generators include:

- Visual Studio 6
- Visual Studio 7
- Visual Studio 7 .NET 2003
- Visual Studio 8 2005
- Visual Studio 9 2008
- Visual Studio 10 2010
- Visual Studio 11 2012
- Visual Studio 12 2013
- Xcode

Extra Generators

These are generators create a configuration to work with an alternative IDE tool and must be included with either an IDE or Command-Line generator.

The supported generators include:

- CodeBlocks
- CodeLite

- Eclipse CDT4
- KDevelop3
- Kate
- Sublime Text 2

Note	In this example ninja is installed via the command `sudo apt-get install ninja-build`

Calling a Generator

To call a CMake generator you can use the `-G` command line switch, for example:

```
cmake .. -G Ninja
```

After doing the above CMake will generate the required Ninja build files, which can be run from using the `ninja` command.

```
$ cmake .. -G Ninja

$ ls
build.ninja  CMakeCache.txt  CMakeFiles  cmake_install.cmake  rules.ninja
```

Building the Examples

Below is sample output from building this example.

```
$ mkdir build.ninja

$ cd build.ninja/

$ cmake .. -G Ninja
-- The C compiler identification is GNU 4.8.4
-- The CXX compiler identification is GNU 4.8.4
-- Check for working C compiler using: Ninja
-- Check for working C compiler using: Ninja -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler using: Ninja
-- Check for working CXX compiler using: Ninja -- 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/J-building-with-ninja/b

$ ninja -v
[1/2] /usr/bin/c++     -MMD -MT CMakeFiles/hello_cmake.dir/main.cpp.o -MF "CMakeFiles/hello_cmake.dir/main.
[2/2] : && /usr/bin/c++      CMakeFiles/hello_cmake.dir/main.cpp.o  -o hello_cmake  -rdynamic && :

$ ls
build.ninja  CMakeCache.txt  CMakeFiles  cmake_install.cmake  hello_cmake  rules.ninja

$ ./hello_cmake
Hello CMake!
```