Introduction to CMake

Peter von Niederhäusern
peter.vonniederhaeusern@bfh.ch

University of Applied Sciences of Bern
Institute for Human Centered Engineering - cpvrLab

18. März 2016

```
cmake_minimum_required(...)
```

Outline

Toolchains

Umbrella term for a collection of tools (compilers, linkers, archivers, debuggers) to produce and maintain code.

- Visual Studio for Windows
- (Cygwin Environment for Windows)
- MinGW Toolchain for Windows
- GCC Toolchain for Linux, OS X
- Xcode Package for OS X
- LLVM Toolchain for Linux, OS X, (Windows)
- ...

Build Systems

buildsysteme => compiling,linking,ect
IDE = inkl. GUI (ide) verwendet build systeme)

Umbrella term for entities that are responsible to invoke, in the correct order, the tools necessary to produce code.

- Difference between IDEs and build systems
- Do you know any build systems?

Build Systems - cont'd

Build systems you may have encountered:

- Java: Apache (Ant | Maven)
- Scala: sbt
- JavaScript: GRUNT
- dlang: DUB
- (MSBuild)

Build Systems - cont'd

Finally, some build systems specific for C++:

- Roll your own (fancy shell scripts)
- GNU Autotools (standard Makefiles):
  `./configure && make && make install`
- Microsoft's NMake
- ...

Build Systems - cont'd

And some more:

- SCons (Python based)
- Waf (Python based)
- Jam (used by Boost)
- QMake (used by Qt)
- ...

THERE IS SOMETHING MISSING IN THIS PIC...
What is it?

Build Systems - cont'd

Wouldn't it be nice to have a tool to generate project files four your beloved IDE?

CMake to the rescue!

- Create solution/project files
  for Visual Studio, Eclipse CDT, Xcode,
  platform specific Makefiles, ...

Framework Issues

- Complex framework setup
- Track framework dependencies
- Setup needs to be correct (LIB, HEADER paths)
- Setup needs to be flexible (different OSes)

Ease of Use

- Multi-platform support
- Support the major IDEs out there
- Fast turnaround (write, compile, link cycle)
- Testing (Continous Integration)
- Documentation
- Packaging & Deployment

CMake - Characteristics

- Meta-Makefile
- Multi-platform support
- Package finding
- Project file generation
- DSL (domain specific language)
- Good documentation

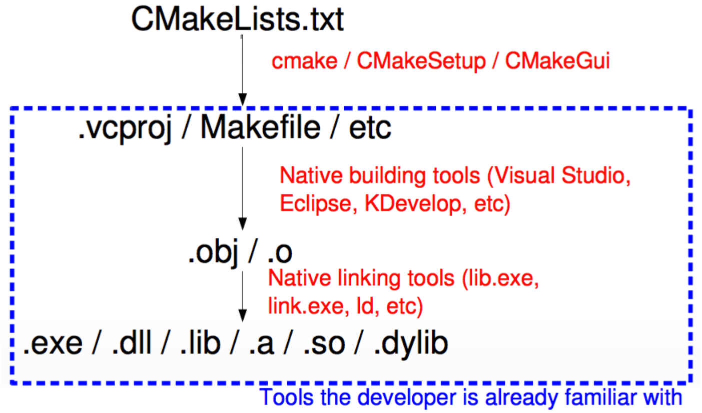
cmake compiliert nicht selbststÄndig -> delegate

vorteil: cmake klein;

CMake - Yet Another Tool

- Windows: download from *www.cmake.org*
- Linux: use the source, `apt-get` or download
- OS X: use `homebrew`, MacPorts or download

CMake - Workflow



CMakeLists.txt
cmake / CMakeSetup / CMakeGui
.vcproj / Makefile / etc
Native building tools (Visual Studio, Eclipse, KDevelop, etc)
.obj / .o
Native linking tools (lib.exe, link.exe, ld, etc)
.exe / .dll / .lib / .a / .so / .dylib
Tools the developer is already familiar with

CMakeLists.txt

```
# bare bones
add_executable(app main.cpp)

# bare bones with flesh
set(SOURCES main.cpp)
add_executable(app ${SOURCES})

# good style
project(helloworld)
set(SOURCES main.cpp)
add_executable(${PROJECT_NAME} ${SOURCES})
```

set => variable wert setzen

${name} derefernzieren der variable

main.cpp

```cpp
// my first "Hello CMake"
#include <iostream>

int main(int argc, char* argv[])
{
   // add your std::cout statement here

   return 0;
}
```

Invocation

To let CMake generate the necessary meta files, either use the command line tool `cmake` or the GUI front ends.

- Windows: `cmake-gui.exe`
- Linux & OS X: `ccmake`

Add a subfolder `'build'` inside your project root from which you will then invoke CMake, cd into `'build'`.

Let CMake generate the build files you want:

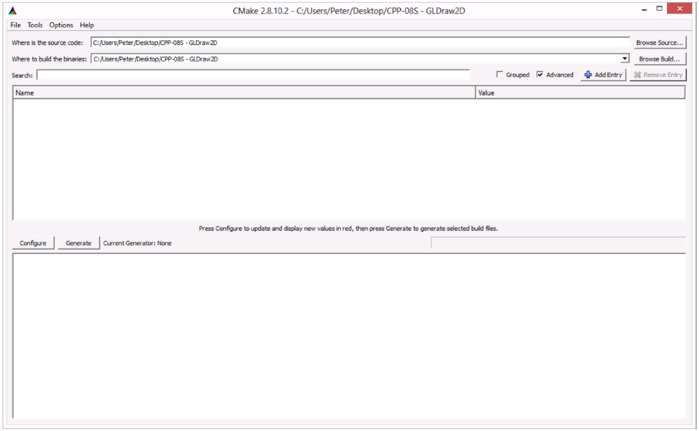
```
cmake -G "NMake Makefiles"..
```

BTW: you can ask CMake for the different generators...

Call the build tool directly (e.g. nmake, make) according to the chosen generator, or simply `cmake --build .` which invokes it for you.

Invocation - cont'd

On Windows, use the GUI front end like this:


CMake 2.8.10.2 - C:/Users/Peter/Desktop/CPP-08S - GLDraw2D

Where is the source code: C:/Users/Peter/Desktop/CPP-08S - GLDraw2D

Where to build the binaries: C:/Users/Peter/Desktop/CPP-08S - GLDraw2D

Grouped [✓ Advanced]

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: None

Or by using the command line tool with `cmake -G .. ..`

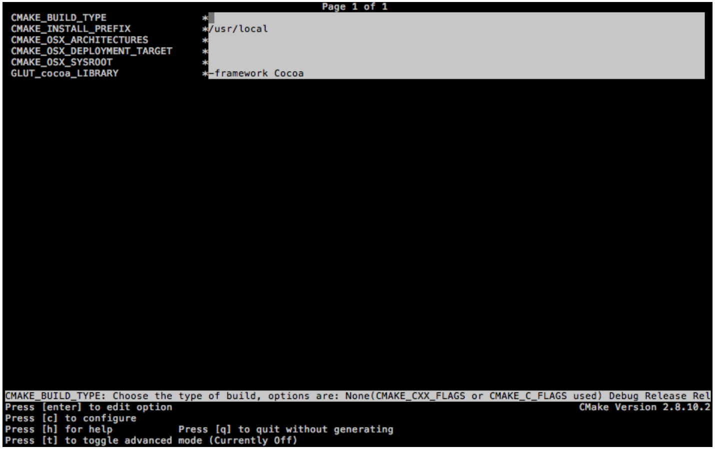
Invocation - cont'd

On Windows:

You might use the Developer Command Prompt for VS2015 to build solution files for the Visual Studio toolchain.

Invocation - cont'd

On Unix, use the GUI front end like this:



```
                                             Page 1 of 1
 CMAKE_BUILD_TYPE                *
 CMAKE_INSTALL_PREFIX            */usr/local
 CMAKE_OSX_ARCHITECTURES         *
 CMAKE_OSX_DEPLOYMENT_TARGET     *
 CMAKE_OSX_SYSROOT               *
 GLUT_cocoa_LIBRARY              *-framework Cocoa

CMAKE_BUILD_TYPE: Choose the type of build, options are: None(CMAKE_CXX_FLAGS or CMAKE_C_FLAGS used) Debug Release Rel
Press [enter] to edit option                                                                 CMake Version 2.8.10.2
Press [c] to configure
Press [h] for help           Press [q] to quit without generating
Press [t] to toggle advanced mode (Currently Off)
```

Or by using the command line tool with `cmake -G .. ..`

Task 1

To get yourself acquainted with CMake, use the information given on these slides to prepare a `CMakeLists.txt` file such that you can generate a Makefile and then an executable for a simple `HelloWorld` application.

Do an out-of-source build.

HAVE FUN!

Visual Studio

Please watch the live instructions.

Eclipse

- Does not support proper multi-project solutions
- Does not support out-of-source builds
- Needs a separate working directory (project folder)

Recipe:

```
mkdir working_dir
cd working_dir
cmake path/to/projects_src -G "Eclipse CDT4 - Unix Makefiles"
```

Let Eclipse import "Existing Projects into Workspace" (point to `working_dir` without copying into the workspaces).

After each added project, rebuild `path/to/projects_src` with CMake and do a refresh on the project layout in Eclipse.

Xcode

Please watch the live instructions.