Introduction to CMake

Projektmanagement im Softwarebereich
OpenMS & SeqAn
CMake – what is it?

Family of tools
CMake → Generates native build environments
CTest → Unit and Suite test system / reporting
CDash → Online reporting system for tests
CPack → Create installers for binary distribution of software

CMake:
Generates native build environments
- UNIX/Linux → Makefiles
- Windows → Visual Studio Projects, NMake,
- Apple → Xcode
Support for Macros
Custom targets/commands
Cross-Platform
OpenSource
Finding/configuring software (Qt, Doxygen, Boost, …)
Who uses it?

SecondLIFE

KDE

OpenMS

SeqAn
Why use a Build System?

You write an application (source code) and need to:

- Compile the source (cross-platform)
- Link to other libraries
- Do compiler specific stuff

You would also love if you were able to:

- Run tests on your software
- Run test of the redistributable package
- See the results of that online (for multiple platforms)
What Build Systems are out there?

**Autotools**
- Autohell
- Bourne shell ('sh'), m4
- Unix platform only (MinGW, CygWin)
- Dependency discovery is mostly manual

**Jam**
- Cross platform
- Not widely used (buggy)

**SCons**

**Bjam (Boost)**

---

Here are some example files for a build system:

- `aclocal.m4` 152.1 KB
- `configure.ac` 21.6 KB
- `configure` 0.7 MB

```plaintext
MAKEDEP_CXX_SUFFIX="|egrep \"\"$(OpenMS_PATH)\|^([^/]*\$)\""
${MAKEDEP_CXX_SUFFIX}"

to escape the damned shell
[] to escape m4
$$ to escape make
```
CMake in Detail

Meta-Build System for:
Visual C++, Kdevelop3, Eclipse, XCode, *makefiles* (Unix, NMake, Borland, Watcom, MinGW, MSYS, Cygwin), Code::Blocks etc → Generator!

Projects are described in CMakeLists.txt
In-Source vs. Out-of-Source

Where to build?

In-source:
- helloapp/hello.cpp
- helloapp/CMakeLists.txt
- helloapp/CMakeCache.txt
- helloapp/hello.exe

Out-of-source:
- helloapp/hello.cpp
- helloapp/CMakeLists.txt
- helloappbuild/CMakeCache.txt
- helloappbuild/hello.exe

Binary tree
CMakeLists.txt

myProject
- hello.cpp
- CMakeLists.txt

PROJECT( helloworld )
ADD_EXECUTABLE( hello hello.cpp )

PROJECT( helloworld )
SET( sources hello.cpp )
ADD_EXECUTABLE( hello ${sources } )
How to add a library? (.dll, .lib, .so, .a)

```cmake
PROJECT( mylibrary )
SET( lib_sources library_1.cpp library_2.cpp )
ADD_LIBRARY( my SHARED ${lib_sources } )
```
CMakeLists.txt - Syntax

# This is a comment
– Commands syntax: COMMAND( arg1 arg2 ... )
– Lists A;B;C # semi-colon separated values
– Variables ${VAR}
– Regular expressions (check CMake FAQ for details...)

foreach (qtlib ${QT_LIBRARIES})
  message(STATUS "Qt lib: ${qtlib}")
endforeach()

WHILE()
  ...
ENDWHILE()

set(STL_DEBUG OFF CACHE BOOL "[GCC only] Enable STL DEBUG mode (very slow).")
if (STL_DEBUG)
  if (CMAKE_COMPILER_IS_GNUCXX)
    add_definitions(/D_GLIBCXX_DEBUG)
    Message(STATUS "STL debug mode: ${STL_DEBUG}")
  else()
    Message(WARNING "STL debug mode is supported for compiler GCC only")
  endif()
endif()

if (WIN32)
  message(STATUS "we're on Windows")
elseif (APPLE)
  message(FATAL_ERROR "Mac is not supported ...")
endif()
Most common commands

SET( VAR value [CACHE TYPE DOCSTRING [FORCE]])
ADD_EXECUTABLE
ADD_LIBRARY
MESSAGE
LIST( APPEND|INSERT|LENGTH|GET|REMOVE_ITEM|REMOVE_AT|SORT ...)
FIND_FILE
FIND_LIBRARY
FIND_PROGRAM
FIND_PACKAGE
EXEC_PROGRAM( bin [work_dir] ARGS <..> [OUTPUT_VARIABLE var] [RETURN_VALUE var] )
OPTION( OPTION_VAR “description string” [initial value] )
CMakeCache.txt

- Populated/Updated during configuration phase
- Contains Entries VAR:TYPE=VALUE
- Speeds up build process
- GUI can be used to change values
- There should be no need to edit it manually!!
As a User…

Create a build directory ("out-of-source-build" concept)
– mkdir OpenMS_build ; cd OpenMS_build

• Configure the package for your system:
  – cmake [options] <source_tree>

  cmake ..\OpenMS
  cmake –G “Visual Studio 9 2008 Win64“ ..\OpenMS

• Build the package:

  make
  devenv #(open Visual Studio)

• Install it:
  – make install
As a User...

Modify your build using CMake Flags

- some are inherent to Cmake
  CMAKE_BUILD_TYPE -- Type of build (Debug, Release, ...)

- some are provided by the software that uses CMake
  STL_DEBUG -- Enable STL Debug mode

cmake -D CMAKE_BUILD_TYPE ="Release" ...
cd $path_to_contrib_build$
cmake -G "$<generator>" "$<path_to_contrib>"

cd $path_to_OpenMS_build$
cmake -D CONTRIB_CUSTOM_DIR:PATH="$<path_to_contrib_build>" -G "$<generator>" "$<path_to_OpenMS>"
CTest

ENABLE_TESTING()
ADD_TEST( testname testexecutable args )
CDash

CDash aggregates, analyzes and displays the results of software testing processes submitted from clients.

For example, build a piece of software on
  Linux, Windows, Mac OS X, Solaris and AIX

Usually, you want two kinds of information:
- Build results on all platforms
- Test (Ctest) results on all platforms

Customizable using XSL
CDash

Site: mirocopbus.ms.fu-berlin.de
Build Name: osx-10.5-gcc-4.2-q47-release
Build Time: 2011-04-11T05:17:03 CEST
Found 39 Errors
Warnings are here

Build Log Line
51

Error
scanning dependencies of target AverageLinkage_test
[ 24%] building CXX object source/TEST/MakeFiles/AverageLinkage_test.dir/AverageLinkage_test.obj
Linking CXX executable bin/AverageLinkage_test
[ 34%] Built target AverageLinkage_test
scanning dependencies of target AveragePosition_test
[ 34%] building CXX object source/TEST/MakeFiles/AveragePosition_test.dir/AveragePosition_test.obj
Linking CXX executable bin/AveragePosition_test
[ 34%] Built target AveragePosition_test
[ 43%] building CXX object CMakeFiles/openMS_GUI.dir/source/VISUAL/Spectrum2DCanvas.C.o
/.../trunk/source/VISUAL/Spectrum2DCanvas.C: In member function 'virtual void OpenMS::Spectrum2DCanvas::contextMenuEvent(QContextMenuEvent*)':
/.../trunk/source/VISUAL/Spectrum2DCanvas.C:2384: error: conversion from 'long unsigned int' to 'const QVariant' is ambiguous

Build Log Line
60

Error
1686-apple-darwin10-4.2.1-g++: CMakeFiles/OpenMS_GUI.dir/source/VISUAL/Spectrum2DCanvas.C.o: No such file or directory
[ 50%] Built target OpenMS_GUI
scanning dependencies of target AxisTickCalculator_test
[ 60%] building CXX object source/TEST/MakeFiles/AxisTickCalculator_test.dir/AxisTickCalculator_test.obj
Cpack

Cpack generates installing packages:
- RPM, DEB, GZip and Bzip2 distributions of both binaries and source code
- NSIS installers (for Microsoft Windows)
- Mac OS X packages (.dmg)

```cmake
add_executable(myexe ${my_src})
install(TARGETS myexe)
install(FILES ${PROJECT_BINARY_DIR}/doc/index.html DESTINATION share/OpenMS/doc COMPONENT doc)
install(DIRECTORY ${PROJECT_BINARY_DIR}/doc/html DESTINATION share/OpenMS/doc COMPONENT doc)
INCLUDE(CPack)
```

cmake -D INSTALL_PREFIX=/usr -D PACKAGE_TYPE=rpm ...
make package
More information

http://www.cmake.org

http://www.elpauer.org/stuff/learning_cmake.pdf