

Das CMake-Buildsystem – Programmentwicklung leichtgemacht

Wolfgang Dautermann

FH JOANNEUM

Chemnitzer Linxstage 2011

- 1 Einleitung
- 2 Hello World Beispiel
- 3 CMake Syntax und Features
- 4 Pakete finden
- 5 CPack - Paketieren von Software
- 6 CTest & CDash

Was ist CMake?

Welcome to CMake, the cross-platform, open-source build system.

Zitat von der CMake-Homepage

CMake is a family of tools designed to build, test and package software. CMake is used to control the software compilation process using simple platform and compiler independent configuration files. CMake generates native makefiles and workspaces that can be used in the compiler environment of your choice.

Wer verwendet CMake?

Bekannte Projekte mit CMake als Buildsystem

- Insight Segmentation and Registration Toolkit (ITK)
- KDE (ab Version 4)
- Mysql
- Second Life
- Scribus
- ...

Hello-World Beispiel

Wir compilieren & installieren ein einfaches C-Programm

CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8)
PROJECT(helloworld)
SET(hello_SRCS helloworld.c)
ADD_EXECUTABLE(helloworld ${hello_SRCS})
INSTALL(TARGETS helloworld RUNTIME DESTINATION bin)
```

In-Source vs. Out-of-source Build

Wo kommen generierte Dateien (Objectfiles, Executables, ...) hin?

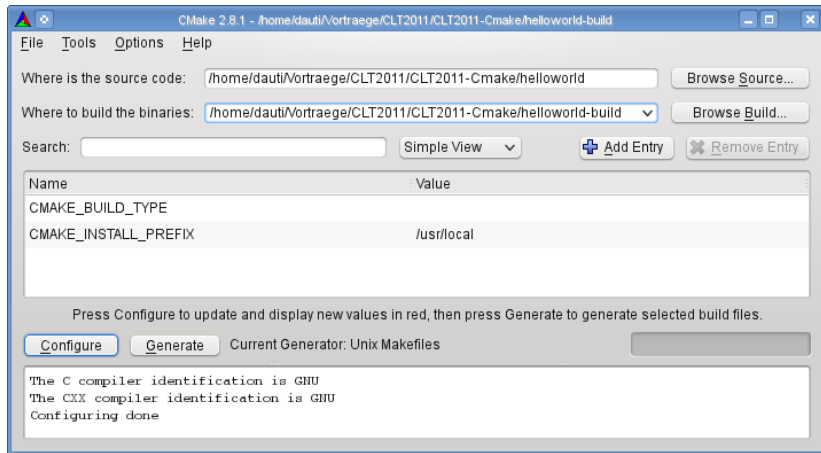
- In-Source: Sourcecode und generierte Dateien (Objectfiles, Executables,...) sind im selben Directory.
Aufräumen (`make clean / make distclean`) notwendig.
- Out-of-Source: Build-Directory \neq Sourcecode-Directory
 - Von CMake supported.
 - (sehr!) empfohlen
 - Sourcecodedirectory wird nicht *verschmutzt*
 - Alle Dateien werden in einem separaten Build-Directory erzeugt.
`make clean: rm -rf *` im Build-directory.
 - Verschiedene Builds (Debug, Release,...) gleichzeitig möglich.

Hello-World Beispiel – Compilieren

Aufruf von cmake

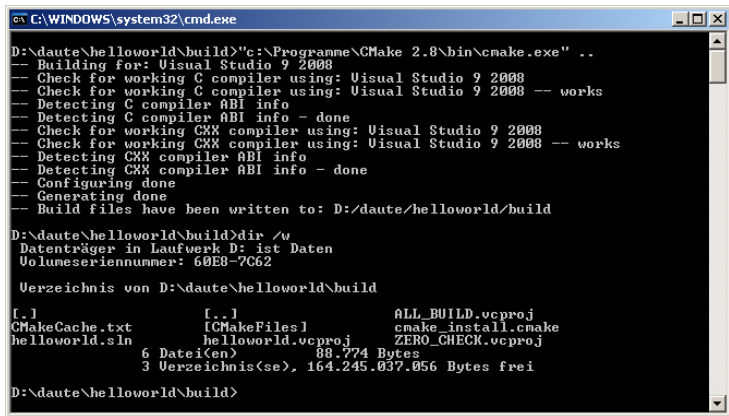
```
~/helloworld-build> cmake ../helloworld
-- The C compiler identification is GNU
-- The CXX compiler identification is GNU
-- Check for working C compiler: /usr/bin/gcc
[...]
-- Configuring done
-- Generating done
-- Build files have been written to: [...]
~/helloworld-build> make [VERBOSE=1]
~/helloworld-build> make install # als root
```

CMake GUI



CMake unter Windows

Erzeugung von Visual-Basic-Projekten



```
C:\WINDOWS\system32\cmd.exe

D:\daute\helloworld\build>"c:\Programme\CMake 2.8\bin\cmake.exe" ..
-- Building for: Visual Studio 9 2008
-- Check for working C compiler using: Visual Studio 9 2008
-- Check for working C compiler using: Visual Studio 9 2008 -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler using: Visual Studio 9 2008
-- Check for working CXX compiler using: Visual Studio 9 2008 -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Configuring done
-- Generating done
-- Build files have been written to: D:/daute/helloworld/build

D:\daute\helloworld\build>dir /w
Datenträger in Laufwerk D: ist Daten
Volumeseriennummer: 60E8-7C62

Verzeichnis von D:\daute\helloworld\build

[.]                [..]                ALL_BUILD.vcproj
CMakeCache.txt     [CMakeFiles]        cmake_install.cmake
helloworld.sln     helloworld.vcproj    ZERO_CHECK.vcproj
                   6 Datei(en)         88.774 Bytes
                   3 Verzeichnis(se), 164.245.037.056 Bytes frei

D:\daute\helloworld\build>
```

CMakeCache.txt

- Daten, die beim Konfigurationslauf gefunden werden (oder z.B. auf der Kommandozeile angegeben werden), kommen in den `CMakeCache.txt` – eine Art Konfigurationsfile.
- Sie können dort ggf. noch geändert werden. (Texteditor oder `make edit_cache`)
- Wenn `CMakeLists.txt` geändert wird, sollte man den Cache löschen und neu aufbauen lassen.

CMake Syntax und Features

Variablen (Case-sensitive! X <> x)

Variablen setzen

```
SET (var wert)
SET (var a.c b.c c.c) # var="a.c;b.c;c.c" (Liste!)
SET (var "hello.c world.c") # var="hello.c world.c"
```

Variablen können beim cmake-Aufruf gesetzt werden:

```
cmake -Dvar=wert ...
```

File globbing

```
FILE(GLOB helloworld_sources *.c )
```

CMake Syntax und Features

CMakeLists.txt in Subdirectories verarbeiten

```
ADD_SUBDIRECTORY (dirname)
```

- gesetzte Variablen werden vererbt!
- In jedem Directory eine eigene Datei CMakeLists.txt

Kontrollstrukturen

if/then/else^a

^aNützliche Beispiele: `if (APPLE)`, `if (WIN32)`, `IF (UNIX)`, `IF (WIN32 AND NOT UNIX)`

```
IF (expression)
...
ELSE (expression)
...
ENDIF (expression)
```

Eine Variable gilt als Wahr, wenn der Variablenwert nicht leer, 0, N, NO, OFF, FALSE, NOTFOUND oder <variable>-NOTFOUND ist.

Kontrollstrukturen

Listen verarbeiten

```
FOREACH(loop_var Liste)  
...  
ENDFOREACH(loop_var Liste)
```

while()-Schleifen

```
WHILE(bedingung)  
...  
ENDWHILE(bedingung)
```

Compilieren: Programme und Libraries

Programme

```
add_executable( <name> sourcefiles )
```

```
add_library(<name> [(STATIC) | SHARED ] sourcefiles)
```

Den Namen ohne OS-spezifische Pre/Suffixes (<name>.exe, <name>.dll, lib<name>.so lib<name>.a,...) angeben – wird automatisch ergänzt (und ist dadurch plattformunabhängig!)

Pakete finden

...ich mag nicht alles selber machen

Pakete finden

```
find_package (<name> [REQUIRED])
```

Bei Bibliotheken werden die folgenden Variablen gesetzt: <name>_FOUND, <name>_LIBRARIES¹, <name>_INCLUDE_DIR².

(Ev. auch noch weitere: `cmake --help-module Find<name>`)

¹manchmal auch <name>_LIBRARY oder <name>_LIBS

²manchmal auch <name>_INCLUDES

Pakete verwenden

Include-Pfad ergänzen

```
include_directories(${<name>_INCLUDE_DIRS})
```

Bibliothek linken

```
target_link_libraries(targetname ${<name>_LIBRARIES})
```

Ggf. Compilerdefinitionen ergänzen (z.B. libPNG)

```
add_definitions(${<name>_DEFINITIONS})
```

Software konfigurieren

Konfigurieren eines Templatefiles

```
configure_file(<input> <output>)
```

- Variablen `${VAR}` oder `@VAR@` werden durch ihre Werte ersetzt,
- `#cmakedefine VAR` wird ersetzt durch `#define VAR` bzw.
`/* #undef VAR */`
- `#cmakedefine01 VAR` wird ersetzt durch `#define VAR 1` bzw.
`#define VAR 0`

Nützlich, um Headerfiles zu konfigurieren, abhängig davon, ob Pakete gefunden wurden oder nicht.

Installationen

Targets installieren

```
install(TARGETS myExe mySharedLib myStaticLib
        RUNTIME DESTINATION bin
        LIBRARY DESTINATION lib
        ARCHIVE DESTINATION lib/static)
```

Files installieren

```
install(FILES files... DESTINATION <dir>)
install(DIRECTORY dir DESTINATION <dir>)
```

Installationspräfix angeben mit: `-DCMAKE_INSTALL_PREFIX:PATH=/my/path`

Cpack - Paketieren von Software

CMake inkludiert CPack, mit dem man Installationspakete in verschiedenen Formaten erstellen kann, z.B.:

Welche Pakete sollen erstellt werden?

```
IF (UNIX)
  SET(CPACK_SOURCE_GENERATOR "TGZ;TBZ2")
  SET(CPACK_GENERATOR "TGZ;TBZ2;DEB;RPM")
ELSE (UNIX)
  SET(CPACK_SOURCE_GENERATOR "ZIP")
  SET(CPACK_GENERATOR "NSIS")
ENDIF (UNIX)
```

CPack - Paketieren von Software

Metadaten festlegen (es gibt noch wesentlich mehr...)

```
# div. Metadaten festlegen:
SET(CPACK_PACKAGE_DESCRIPTION_SUMMARY
    "Description of Helloworld")
SET(CPACK_PACKAGE_VENDOR "The Helloworld Team")
SET(CPACK_PACKAGE_DESCRIPTION_FILE
    "${CMAKE_CURRENT_SOURCE_DIR}/readme.txt")
SET(CPACK_RESOURCE_FILE_LICENSE
    "${CMAKE_CURRENT_SOURCE_DIR}/license.txt")
SET(CPACK_PACKAGE_VERSION_MAJOR "0")
SET(CPACK_PACKAGE_VERSION_MINOR "1")
SET(CPACK_PACKAGE_CONTACT
    "Helloworld Team <helloworldteam@example.org>")
SET(CPACK_PACKAGE_SECTION "games")
INCLUDE(CPack)
```

make package bzw. make package_source erstellt dann die Pakete. (Live Demo)

CTest - Testen von Software

ermöglicht automatisierte Tests

ENABLE_TESTING() und ADD_TEST()

```
ENABLE_TESTING ()  
ADD_TEST (Testname1  
          ${EXECUTABLE_OUTPUT_PATH}/<name> [argumente])  
ADD_TEST (Testname2  
          ${EXECUTABLE_OUTPUT_PATH}/<name> [argumente])  
[...]
```

make test startet dann die Tests (Live-Demo).

Links und weiterführende Infos

- <http://www.cmake.org>
- <http://www.cmake.org/Wiki/CMake>

inkludierte Hilfe

```
man cmake
cmake --help
      --help-full
      --help-command cmd
      --help-module module
      [...]
```

Fragen? Feedback?

Vielen Dank für Ihre Aufmerksamkeit

Wolfgang Dautermann

wolfgang.dautermann [AT] fh-joanneum.at