Cheatography

CMake Cheat Sheet

by kimpel70 via cheatography.com/4828/cs/3368/

Einfache CMake Datei

```
# CMAKE
cmake_minimum_required (VERSION 2.6)
cmake_policy(SET CMP0014 NEW)
cmake_policy(SET CMP0015 NEW)
PROJECT(hello)
set(PROJECT_SRC main.cpp)
add_executable(${PROJECT_NAME}
${PROJECT_SRC})
```

Executable erstellen

Beispiel exe + lib erstellen

CMakeLists.txt im Hauptverzeichnis

```
PROJECT(example-2)
add_subdirectory(lib)
add_subdirectory(exe)
```

CMakeLists.txt im Verzeichnis *lib*

```
PROJECT(lib)
set(PROJECT_SRC lib.cpp)
add_library(${PROJECT_NAME}
${PROJECT_SRC})
```

CMakeLists.txt im Verzeichnis *exe*

```
PROJECT(hello)
set(PROJECT_SRC main.cpp)
include_directories(../lib)
add_executable(${PROJECT_NAME}
${PROJECT_SRC})
target_link_libraries(${PROJECT_NAME} lib)
```

Die folgenden Zeilen wurden am Anfang der Skripte zwecks Übersichtlichkeit weggelassen:

```
cmake_minimum_required (VERSION 2.6)
cmake_policy(SET CMP0014 NEW)
cmake_policy(SET CMP0015 NEW)
```

CMake -E für plattformunabhängige Kommandos

chdir, compare_files, copy, copy_directory, copy_if_different, echo, echo_append, environment, make_directory, md5sum, remove, remove_directory, rename, tar, time, touch, touch_nocreate.

Windows: comspec, delete_regv, write_regv.
UNIX: create_symlink.

Cmake --build

Projekt erstellen mit dem konfiguriertem Tool für diese Platform

Variabel Scope

Variabel mit *SET* werden zwar an Projekte die mit add_directory aufgerufen vererbt, aber die Variabeln in den Unterprojekten nicht automatisch an den
Aufrufer übergeben.
Das kann per SET(VAR "22" PARENT_SCOPE) erreicht werden. **Aber Achtung:** Es wird nur in den direkten*parent* eingebunden, es kann natürlich mehrere Stufen von *add_directory* geben

Generator Expressions

```
$<$<CONFIG:Debug>:libeay32d.lib>$
```

Bibliotheksname mit suffix wenn Debug

```
$<$<CONFIG:Release>:libeay32.lib>$
```

Bibliotheksname ohne suffix wenn Release
Kommandos: **target_link_libraries(), target_include_directories(), target_compile_definitions()**

Achtung: Die Definitionen können auch mit **SET** verwendet werden, dann dürfen keine Leerzeichen zwischen den *Generator Expressions* stehen, andernfalls wird eine Liste daraus, die nicht mehr das gewünschte Ergebnis liefert.
Auswertung erfolgt dann aber erst beim Einsetzen in einer der oben genannten Kommandos!

zu CMake konvertierte Projekte

Xerces(Wrapper)	zlib	freetype
netpgp	openssl	

Als *Subrepository* ins Projekt aufnehmen und im **CMakeLists.txt** mit *add_subdirectory()* einbinden **FERTIG!**

add_custom_command PRE_BUILD

Nicht auf Unix verfügbar, damit nicht plattformunabhängig!
Besser:
add_custom_command(OUTPUT <Abhängigkeit>)
und *Abhängigkeit* in SRC definieren
VERBATIM nicht vergessen!

build_support (repositoryid: _230)

Utility/GetCorrespondingHeaders	zu Liste von CPP-Dateien Header ermitteln
FindOracle	Oracle Installation finden

wiederverwendbare CMake-Dateien (per include einbinden)

Superbuild

Es gibt Situationen in den quasi ein CMake zwingend vor einem anderen Laufen muß, um die benötigten Abhängigkeiten bereitzustellen.
Standardweg:
Das Superbuild CMakeLists.txt enthält ausschließlich externe Projekte, für die Abhängigkeiten. Zu guter letzt wird das eigentliche Projekt selbst als externes Projekt eingebunden.
Sind die Abhängigkeiten bereitgestellt, kann einfach auf dem Projekt-CMake aufgesetzt werden, das Superbuild-CMake muß nicht weiter beachtet werden