Lecture 3. CMake Project Build System

Cross-Platform Application Development

September 22, 2017

Functionality Outline

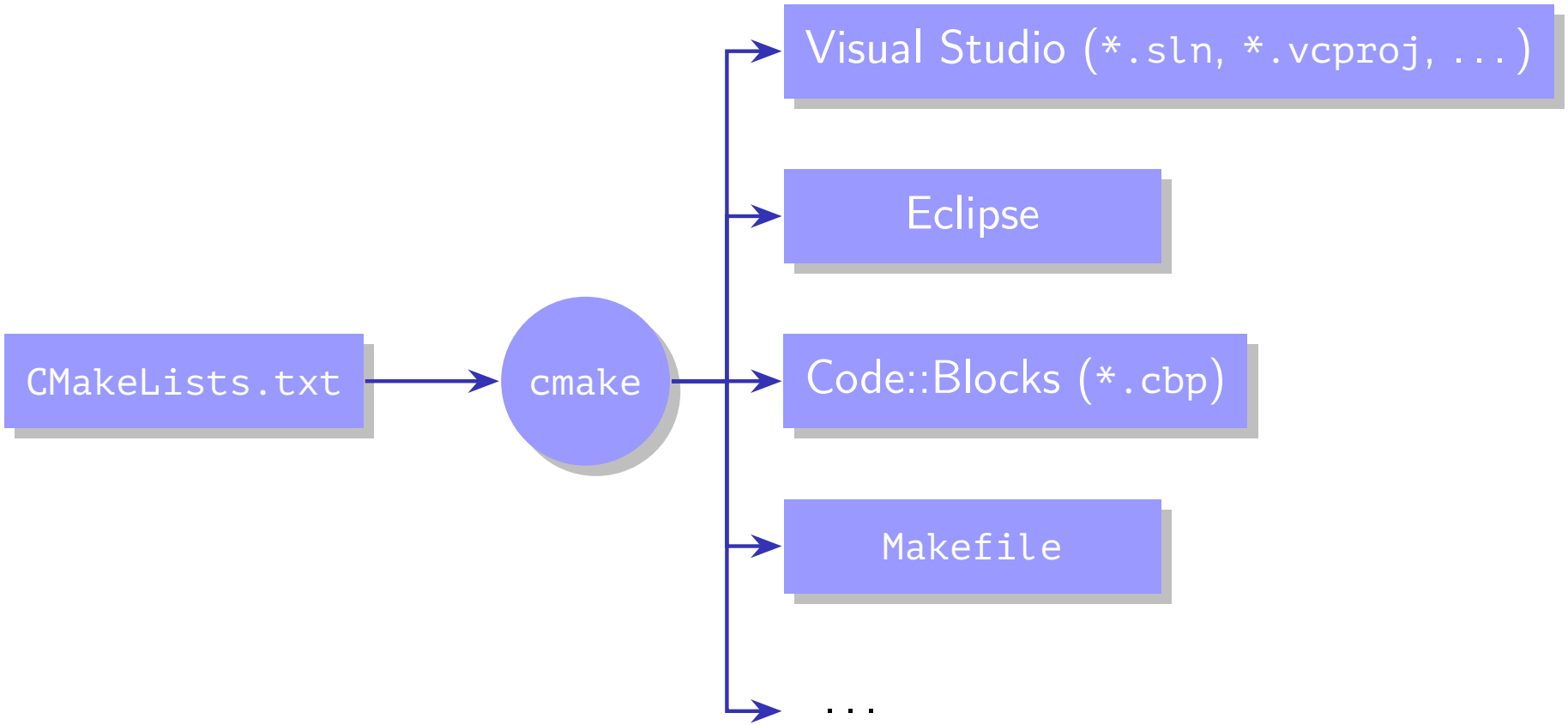

Figure 1: project generation with CMake system

Project Structure

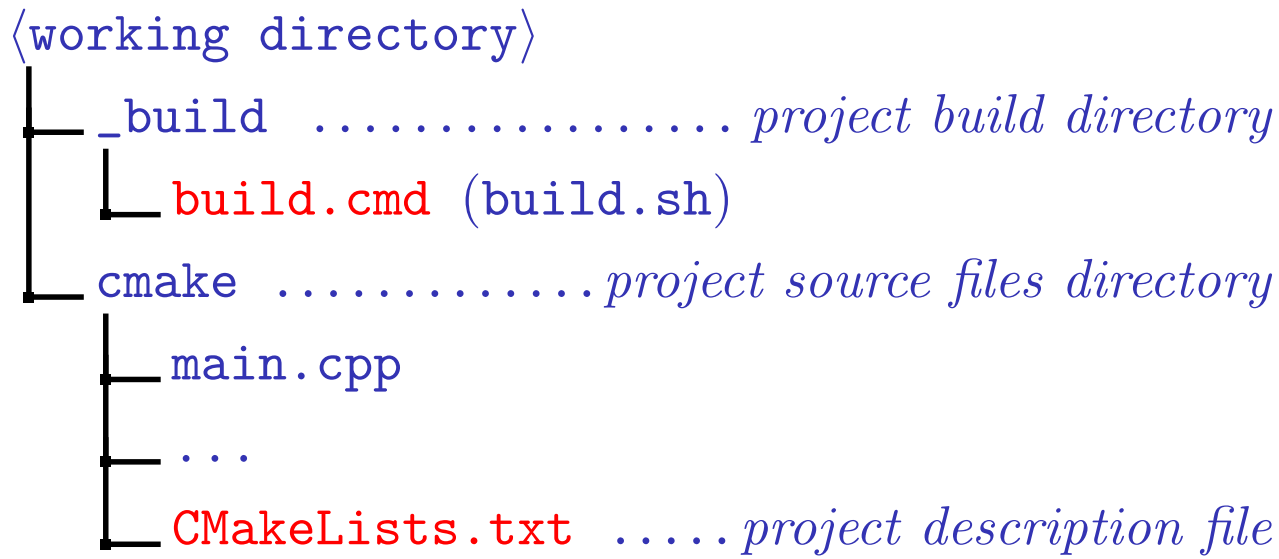

Figure 2: structure of a simple project directory

Example

Example (build.cmd, using PATH environment variable)

```
@set PATH=C:\Program Files (x86)\CodeBlocks\MinGW\bin;%PATH%

cmake -G "CodeBlocks - MinGW Makefiles" ../test_cmake
```

Example (build.cmd, using CMAKE_PREFIX_PATH CMake variable)

```
@set PATH=C:\Qt\Qt5.7.0\Tools\mingw530_32\bin;%PATH%

cmake^
  -G "MinGW Makefiles"^
  -D CMAKE_PREFIX_PATH="C:\Qt\Qt5.7.0\5.7\mingw53_32"^
  ..\qt-examples-2
```

Example

Example (CMakeLists.txt)

```
# CMakeLists.txt for a simple project

project(test_cmake)

add_executable(test_cmake main.cpp)

# End of File
```

Example (cont.)

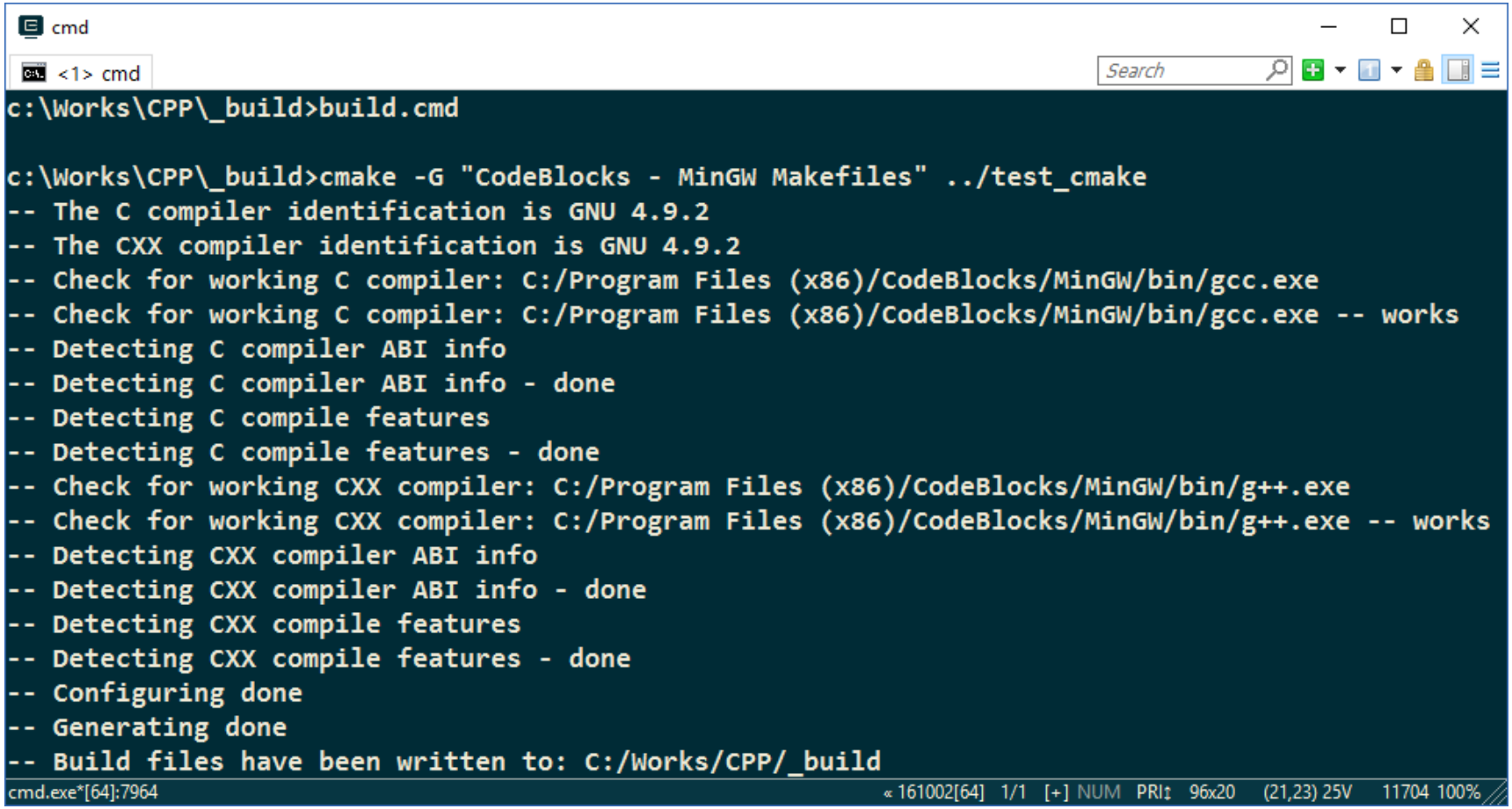

Figure 3: Output of CMake Program

Example (end)

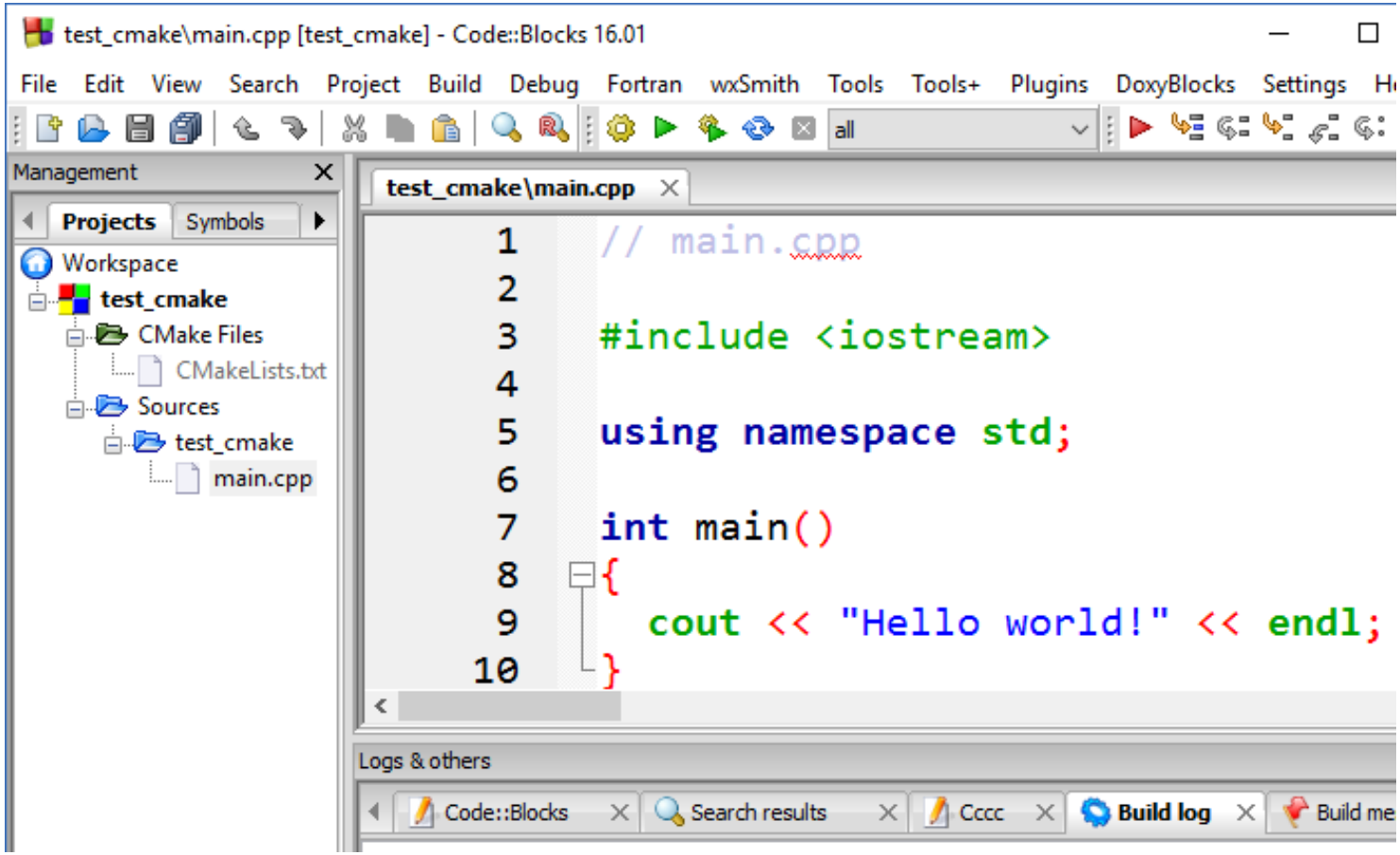

Figure 4: Code::Blocks IDE with an Opened Generated Project

Project Sreucture

```
⟨working directory⟩
├── _build_project ..... project build directory
│   └── build.cmd (build.sh)
└── project ..... project source files directory
    ├── sample_lib ..... for the library
    │   ├── sample_module.cpp
    │   ├── sample_module.h
    │   └── CMakeLists.txt
    ├── sample_program ..... for the executable
    │   ├── main.cpp
    │   └── CMakeLists.txt
    └── CMakeLists.txt ..... project description file
```

Figure 5: structure of a directory for project with a library

Example

Example (CMakeLists.txt)

```
cmake_minimum_required(VERSION 2.8)

add_subdirectory(sample_lib)
add_subdirectory(sample_program)
```

Example

Example (CMakeLists.txt)

```
cmake_minimum_required(VERSION 2.8)

add_subdirectory(sample_lib)
add_subdirectory(sample_program)
```

Example (sample_lib/CMakeLists.txt)

```
project(sample_lib)

add_library(sample_lib sample_module.cpp sample_module.h)
```

Example

Example (sample_program/CMakeLists.txt)

```
project(sample_program)

add_executable(sample_program main.cpp)

include_directories(../sample_lib)

target_link_libraries(sample_program sample_lib)
```

Example

Example (build.cmd)

```
cmake -G "CodeBlocks - MinGW Makefiles" -DBUILD_SHARED_LIBS=0 ../project
```

Example

Example (build.cmd)

```
cmake -G "CodeBlocks - MinGW Makefiles" -DBUILD_SHARED_LIBS=0 ../project
```

Example (sample_lib/CMakeLists.txt)

```
set(BUILD_SHARED_LIBS FALSE)
```

Example

Example (build.cmd)

```
cmake -G "CodeBlocks - MinGW Makefiles" -DBUILD_SHARED_LIBS=0 ../project
```

Example (sample_lib/CMakeLists.txt)

```
set(BUILD_SHARED_LIBS FALSE)
```

Example (sample_lib/CMakeLists.txt)

```
project(sample_lib)

add_library(sample_lib STATIC sample_module.cpp sample_module.h)
```

Standard Variables

CMAKE_INCLUDE_PATH	CMAKE_LIBRARY_PATH	CMAKE_PROGRAM_PATH
CMAKE_FRAMEWORK_PATH	CMAKE_APPBUNDLE_PATH	CMAKE_PREFIX_PATH
CMAKE_INSTALL_PREFIX	BUILD_SHARED_LIBS	

Table 1: influencing CMake behavior

CYGWIN	MSVC_VERSION:	1200, 1300, ..., 1900
MINGW	CMAKE_COMPILER_IS_GNUCC	
MSVC	CMAKE_COMPILER_IS_GNUCXX	
MSVC80	UNIX	
MSVC_IDE	WIN32	

Table 2: describing the system

Standard Variables (end)

CMAKE_BINARY_DIR	CMAKE_CURRENT_BINARY_DIR
CMAKE_SOURCE_DIR	CMAKE_CURRENT_SOURCE_DIR

Table 3: for information

CMAKE_INCLUDE_CURRENT_DIR	CMAKE_LIBRARY_OUTPUT_DIRECTORY
CMAKE_RUNTIME_OUTPUT_DIRECTORY	CMAKE_ARCHIVE_OUTPUT_DIRECTORY

Table 4: managing the build process

Example

Example (CMakeLists.txt)

```
project(my_project)

set(BINARY_DIR "${CMAKE_BINARY_DIR}")
set(CMAKE_RUNTIME_OUTPUT_DIRECTORY "${BINARY_DIR}/bin")
set(CMAKE_LIBRARY_OUTPUT_DIRECTORY "${BINARY_DIR}/lib")
set(CMAKE_ARCHIVE_OUTPUT_DIRECTORY "${BINARY_DIR}/lib")

add_subdirectory(my_library_1)
add_subdirectory(my_library_2)
add_subdirectory(my_program)
# ...
```

Commands for Adding Targets

Adding Targets

add_executable(
⟨*logical_target_name*⟩ [WIN32]
⟨*source_unit*$_1$⟩ ... ⟨*source_unit*$_n$⟩)

add_library(
⟨*logical_target_name*⟩ [STATIC | SHARED | MODULE]
⟨*source_unit*$_1$⟩ ... ⟨*source_unit*$_n$⟩)

add_subdirectory(
⟨*subproject_directory*⟩ [⟨*build_subdirectory*⟩])

Commands for Adding Targets (end)

Adding Targets (end)

add_library(
⟨*logical_target_name*⟩
⟨*library_type*⟩
IMPORTED)

⟨*library_type*⟩ ::= SHARED | STATIC | MODULE | UNKNOWN

Commands for Setting Project Subdirectories

Settings

include_directories(
  ⟨*directory*$_1$⟩ ... ⟨*directory*$_n$⟩)

add_definitions(
  ⟨*definition*$_1$⟩ ... ⟨*definition*$_n$⟩)

add_compile_options(
  ⟨*setting*$_1$⟩ ... ⟨*setting*$_n$⟩)

Examples

```
include_directories(
  include)

add_definitions(
  -DDEBUG -DEXTRA_TESTS=4)

add_compile_options(
  -std=c++11)
```

Commands for Setting Targets

Commands

target_link_libraries(
  ⟨*target_name*⟩
  [⟨$library_1$⟩ ... ⟨$library_n$⟩])

⟨*command_name*⟩(
  ⟨*target_name*⟩
  INTERFACE | PUBLIC | PRIVATE
    [⟨$setting_{1,1}$⟩ ... ⟨$setting_{1,m}$⟩]
  [
    INTERFACE | PUBLIC | PRIVATE
      [⟨$setting_{2,1}$⟩ ... ⟨$setting_{2,n}$⟩]
    ...
  ])

Commands

⟨*command_name*⟩ ::=
  target_include_directories |
  target_compile_definitions |
  target_compile_options |
  target_compile_features |
  target_link_libraries

Example of Target Transitive Settings Use

Example (library)

```
add_library(
  my_lib
  my_lib.cpp my_lib.h)

target_include_directories(
  my_lib
  INTERFACE .)
```

Example (executable)

```
add_executable(
  my_prog
  my_prog.cpp)

target_link_libraries(
  my_prog my_lib)
```

Example of Target Transitive Settings Use

Example (library)

```
add_library(
  my_lib
  my_lib.cpp my_lib.h)

set(
  CMAKE_INCLUDE_CURRENT_DIR_IN_INTERFACE
  ON)
```

Example (executable)

```
add_executable(
  my_prog
  my_prog.cpp)

target_link_libraries(
  my_prog my_lib)
```

Example of Setting Compilation Features

Example (ex-cpp14.cpp, C++14)

```
#include <iostream>

int main()
{
  auto n = 0b0'0100'1011;
  std::cout << n << std::endl;
}
```

Example (CMakeLists.txt)

```
cmake_minimum_required(VERSION 3.2.0)

project(ex-cpp14)

add_executable(ex-cpp14 ex-cpp14.cpp)

target_compile_features(
  ex-cpp14
  PRIVATE                    # or:
    cxx_auto_type            # cxx_std_14
    cxx_binary_literals
    cxx_digit_separators)
```

Commands for Variable Manipulation

Value Assignment

set(
  ⟨*variable_name*⟩ ⟨*value*⟩
  [[CACHE ⟨*type*⟩ ⟨*description_string*⟩] | PARENT_SCOPE])

⟨*type*⟩ ::=
  FILEPATH | PATH | STRING | BOOL | INTERNAL

option(
  ⟨*variable_name*⟩ ⟨*description_string*⟩ [ON | OFF])

Commands for Variable Manipulation (end)

Value Assignment (end)

set(⟨*variable_name*⟩)

unset(⟨*variable_name*⟩ [CACHE | PARENT_SCOPE])

set(
⟨*variable_name*⟩ ⟨$value_1$⟩ ... ⟨$value_n$⟩)

math(EXPR ⟨*variable_name*⟩ ⟨*expression*⟩)

Properties

Entities

- (Sub)project Directories;
- Targets;
- Tests;
- Source Files;
- Cache Variables;
- Installation Files.

Command to Retrieve Property Value

Command

get_property(
⟨*variable_name*⟩
⟨*entity*⟩
PROPERTY ⟨*property_name*⟩
[SET | DEFINED])

Command

⟨*entity*⟩ ::=
GLOBAL |
DIRECTORY [⟨*directory*⟩] |
TARGET ⟨*target_name*⟩ |
SOURCE ⟨*file*⟩ |
INSTALL ⟨*file*⟩ |
TEST ⟨*test_name*⟩ |
CACHE ⟨*variable_name*⟩ |
VARIABLE

Command to Set Properties

Properties Setting

set_property(
⟨*entities*⟩
[APPEND] [APPEND_STRING]
PROPERTY ⟨*property_name*⟩
[⟨$value_1$⟩ ... ⟨$value_m$⟩])

Command to Set Properties (end)

Properties Setting (end)

⟨*entities*⟩ ::=
  GLOBAL |
  DIRECTORY [⟨*directory*⟩] |
  TARGET [⟨*target_name*$_1$⟩ ... ⟨*target_name*$_n$⟩] |
  SOURCE [⟨*file*$_1$⟩ ... ⟨*file*$_n$⟩] |
  INSTALL [⟨*file*$_1$⟩ ... ⟨*file*$_n$⟩] |
  TEST [⟨*test_name*$_1$⟩ ... ⟨*test_name*$_n$⟩] |
  CACHE [⟨*variable_name*$_1$⟩ ... ⟨*variable_name*$_n$⟩]

Example

Example (CMakeLists.txt)

```
add_executable(exec main.cpp file1.cpp file1.h)

set_property(
  TARGET exec
  PROPERTY OUTPUT_NAME
  my_prog)
```

Commands of Flow Control

Branching

if(⟨*condition*$_1$⟩)
  ⟨*commands*⟩
[**elseif**(⟨*condition*$_2$⟩)
  ⟨*commands*⟩
...]
[**else**()
  ⟨*commands*⟩]
endif()

Commands of Flow Control (cont.)

Branching (end)

⟨*condition*⟩ ::=
⟨*variable*⟩ | (⟨*condition*⟩) | NOT ⟨*condition*⟩ |
⟨*condition*⟩ AND ⟨*condition*⟩ | ⟨*condition*⟩ OR ⟨*condition*⟩ |
EXISTS ⟨*file_or_directory*⟩ | IS_DIRECTORY ⟨*path*⟩ |
⟨*variable_or_value*⟩ LESS ⟨*variable_or_value*⟩ |
⟨*variable_or_value*⟩ GREATER ⟨*variable_or_value*⟩ |
⟨*variable_or_value*⟩ EQUAL ⟨*variable_or_value*⟩ |
⟨*variable_or_value*⟩ STRLESS ⟨*variable_or_value*⟩ |
⟨*variable_or_value*⟩ MATCHES ⟨*regular_expression*⟩ |
...

Example

Example (CMakeLists.txt)

```
if(MSVC AND MSVC_VERSION GREATER 1400)
  add_definitions(/MP)
endif()
```

Commands of Flow Control (cont.)

Iterating Values

foreach(⟨*variable_name*⟩ ⟨*value*$_1$⟩ ... ⟨*value*$_n$⟩)
  ⟨*commands*⟩
endforeach()

foreach(⟨*variable_name*⟩ IN
  [LISTS ⟨*variable_name*$_1$⟩ ... ⟨*variable_name*$_n$⟩]
  [ITEMS ⟨*value*$_1$⟩ ... ⟨*value*$_n$⟩])

foreach(⟨*variable_name*⟩ RANGE ⟨*maximum*⟩)

foreach(⟨*variable_name*⟩ RANGE ⟨*start*⟩ ⟨*end*⟩ [⟨*step*⟩])

Command for Including a File or Module

`include()` Command

include(
  ⟨*file*⟩ | ⟨*module*⟩
  [OPTIONAL] [RESULT_VARIABLE ⟨*variable_name*⟩])

Example

Example (CMakeLists.txt)

```
cmake_minimum_required(VERSION 2.8)

set(
  SUBPROJECTS
  program1 program2 super_program)

include(build.cmake)
```

Example (end)

Example (build.cmake)

```
foreach(PROJ ${SUBPROJECTS})
  set(
    MY_BUILD_${PROJ} TRUE
    CACHE BOOL "Build the ${PROJ} subproject")
  if(MY_BUILD_${PROJ} AND
    EXISTS
      "${CMAKE_SOURCE_DIR}/${PROJ}/CMakeLists.txt")
    message(STATUS "The project ${PROJ} will be included")
    add_subdirectory(${PROJ})
  else()
    message(STATUS "The project ${PROJ} will NOT be included")
  endif()
endforeach()
```

Example

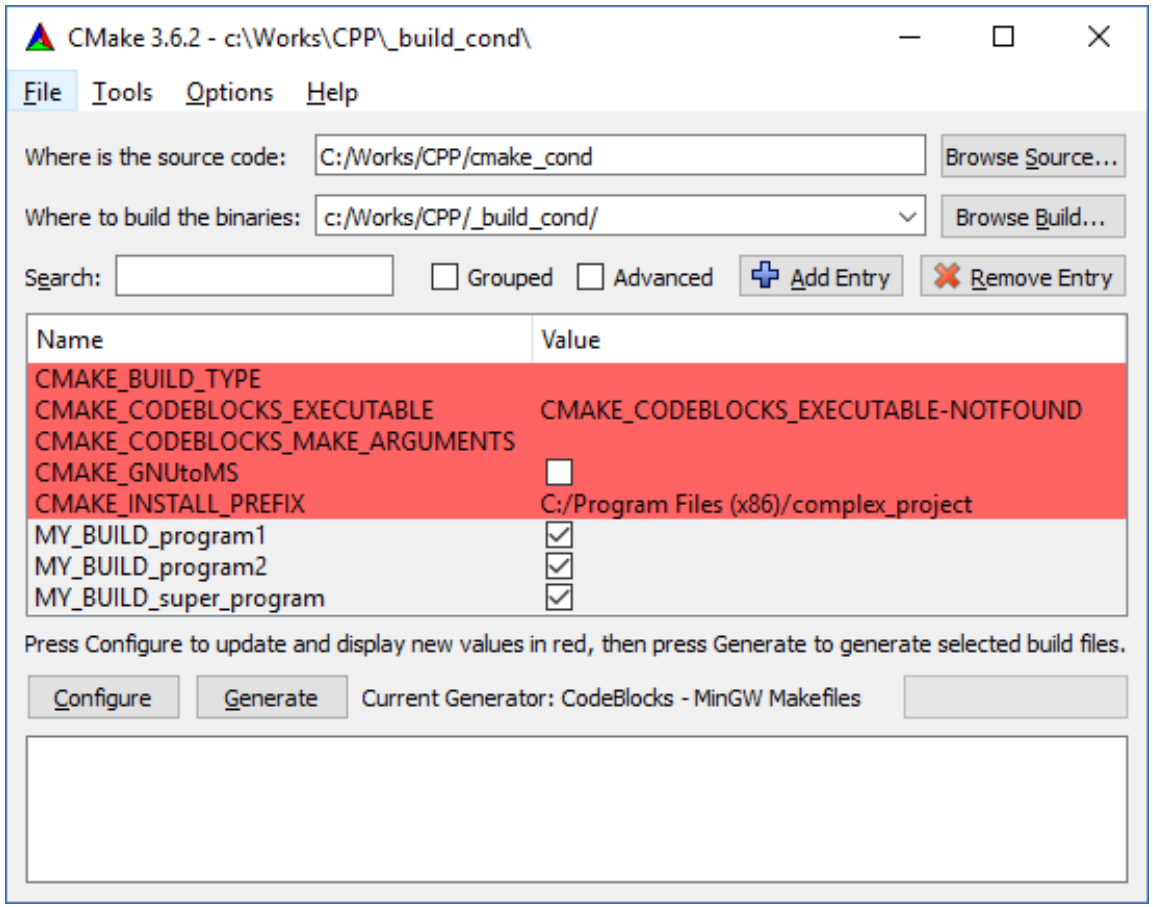

Figure 6: Project Generating with ccmake Utility

Commands of Flow Control (end)

Loop

while(⟨*condition*⟩)
  ⟨*commands*⟩
endwhile()

break()

function(⟨*function_name*⟩ [⟨*paramater*$_1$⟩ ... ⟨*paramater*$_n$⟩])
  ⟨*commands*⟩
endfunction()

ARGC	ARGV0, ARGV1, ...	ARGV	ARGN

Table 5: variables to access function parameters

Example

Example (build.cmake)

```
function(my_create_projects BASE_NAME)
  foreach(PROJ ${ARGN})
    # ...
  endforeach()
endfunction()

# ...

my_create_projects(
  "my_application"
  project1 project2 superproject)
```

Command for Parsing a Path

`get_filename_component()` Command

`get_filename_component``(`
  ⟨*variable_name*⟩ ⟨*path_to_file*⟩ ⟨*component*⟩
  [`CACHE`]`)`

⟨*component*⟩ `::=`
  `DIRECTORY` | `NAME` | `EXT` | `NAME_WE` | `ABSOLUTE` | `REALPATH`

Commands for File Searching

find_file(), **find_library**(), **find_path**(), **find_program**() Commands

⟨*command_name*⟩(
  ⟨*variable_name*⟩
  ⟨*alternative_names*⟩
  [HINTS ⟨$path_{1,1}$⟩ ... ⟨$path_{1,m}$⟩ [ENV ⟨$env\_variable_1$⟩]]
  [PATHS ⟨$path_{2,1}$⟩ ... ⟨$path_{2,p}$⟩ [ENV ⟨$env\_variable_2$⟩]]
  [PATH_SUFFIXES ⟨$suffix_1$⟩ ... ⟨$suffix_k$⟩]
  [DOC ⟨*documentation_string*⟩])

⟨*command_name*⟩ ::=
  find_file | **find_library** | **find_path** | **find_program**

⟨*alternative_names*⟩ ::=
  ⟨*name*⟩ | NAMES ⟨$name_1$⟩ ... ⟨$name_n$⟩

Search Directories

Command	Path from CMAKE_PREFIX_PATH	Path 2
find_file(), **find_path**()	⟨path⟩/include	CMAKE_INCLUDE_PATH, CMAKE_FRAMEWORK_PATH
find_library()	⟨path⟩/lib	CMAKE_LIBRARY_PATH, CMAKE_FRAMEWORK_PATH
find_program()	⟨path⟩/bin, ⟨path⟩/sbin	CMAKE_PROGRAM_PATH, CMAKE_APPBUNDLE_PATH

Table 6: search directories for commands

Library Import

Example

```
find_path(MPIR_H_DIR mpir.h)
if(NOT MPIR_H_DIR)
  message(
    SEND_ERROR
    "Could not find mpir.h")
endif()
find_library(MPIR_LIB mpir)
if(NOT MPIR_LIB)
  message(
    SEND_ERROR
    "Could not find mpir")
endif()
```

Example (end)

```
add_library(mpir STATIC IMPORTED)
set_property(
  TARGET mpir
  PROPERTY
    INTERFACE_INCLUDE_DIRECTORIES
  ${MPIR_H_DIR})
set_property(
  TARGET mpir
  PROPERTY IMPORTED_LOCATION
  ${MPIR_LIB})
# ...
target_link_libraries(my_prog mpir)
```

Command for Searching a Package

`find_package()` Command

`find_package``(`
⟨*package_name*⟩ [⟨*version*⟩] [`EXACT`] [`QUIET`] [`REQUIRED`]
[[`COMPONENTS`] [⟨*component*$_1$⟩ ... ⟨*component*$_n$⟩]]
[`CONFIG` | `NO_MODULE`]
[`NAMES` ⟨*name*$_1$⟩ ... ⟨*name*$_n$⟩])

Example

Example (build.cmake)

```
cmake_minimum_required(VERSION 2.8)

project(test_opencv)

find_package(OpenCV REQUIRED core highgui imgproc)

add_executable(test_opencv WIN32 test_opencv.cpp)
include_directories(${OpenCV_INCLUDE_DIRS})
target_link_libraries(test_opencv ${OpenCV_LIBS})
```

Command to Install Targets

`install()` Command

```
install(
  TARGETS ⟨target_name_1⟩ ... ⟨target_name_n⟩
  [
    [ ARCHIVE | LIBRARY | RUNTIME ]
    [ DESTINATION ⟨directory⟩ ]
    [ CONFIGURATIONS [ Debug | Release | ... ] ]
    [ COMPONENT ⟨component_name⟩ ]
  ]
  ...)
```

Command to Install Files

`install()` Command (cont.)

```
install(
  FILES | PROGRAMS ⟨file_1⟩ ... ⟨file_n⟩
  DESTINATION ⟨directory⟩
  [ CONFIGURATIONS [ Debug | Release | ... ] ]
  [ COMPONENT ⟨component_name⟩ ]
  [ RENAME ⟨name⟩ ])
```

Command to Install Directories

`install()` Command (end)

```
install(
  DIRECTORY [ ⟨directory_1⟩ ... ⟨directory_n⟩ ]
  DESTINATION ⟨directory⟩
  [ CONFIGURATIONS [ Debug | Release | ... ] ]
  [ COMPONENT ⟨component_name⟩ ]
  [ FILES_MATCHING ]
  [
    [ PATTERN ⟨pattern⟩ | REGEX ⟨regular_expression⟩ ]
    [ EXCLUDE ]
  ]
  [ ... ])
```

Library Installation

Example

```
add_library(
  my_library_1
  f.cpp f.h)

get_property(
  LIB_TYPE
  TARGET my_library_1
  PROPERTY TYPE)
```

Example (cont.)

```
if(LIB_TYPE STREQUAL SHARED_LIBRARY)
  install(
    TARGETS my_library_1
    COMPONENT user
    RUNTIME
      DESTINATION bin
    LIBRARY
      DESTINATION lib)
endif()
```

Library Installation (end)

Example (cont.)

```
install(
  TARGETS my_library_1
  COMPONENT developer
  RUNTIME
    DESTINATION bin
  LIBRARY
    DESTINATION lib
  ARCHIVE
    DESTINATION lib)
```

Example (end)

```
install(
  DIRECTORY .
  DESTINATION include
  COMPONENT developer
  FILES_MATCHING
    PATTERN "*.h")
```

A Script for Building and Installing a Library

Example (build.cmd)

```
cmake^
  -G "MinGW Makefiles"^
  -D CMAKE_INSTALL_PREFIX=D:\Install\my_library_1^
  D:\Work\Source\my_library_1

mingw32-make

cmake -D COMPONENT=developer -P cmake_install.cmake
```

Command to Add a Phony Target

`add_custom_target()` Command

`add_custom_target``(`
⟨*logical_target_name*⟩
[ALL]
[⟨*path_to_command*$_1$⟩ [⟨*argument*$_{1,1}$⟩ ... ⟨*argument*$_{1,m}$⟩]]
[
COMMAND
⟨*path_to_command*$_2$⟩ [⟨*argument*$_{2,1}$⟩ ... ⟨*argument*$_{2,n}$⟩]
...
]
[DEPENDS ⟨*file*$_1$⟩ ... ⟨*file*$_k$⟩]
[WORKING_DIRECTORY ⟨*directory*⟩]
[VERBATIM]
[SOURCES ⟨*source_file*$_1$⟩ ... ⟨*source_file*$_p$⟩])

Example

Example (7zip.cmake)

```
set(BINDIR32_ENV_NAME "ProgramFiles(x86)")

find_program(
  7ZIP_EXECUTABLE
  NAMES
    7z 7za
  PATHS
    "$ENV{ProgramFiles}/7-Zip"
    "$ENV{${BINDIR32_ENV_NAME}}/7-Zip"
    "C:/Program Files/7-Zip"
    "C:/Program Files (x86)/7-Zip"
  )
```

Example (end)

Example (7zip.cmake, end)

```
if(7ZIP_EXECUTABLE)
  add_custom_target(
    create_archive
    COMMAND
      "${7ZIP_EXECUTABLE}"
        a "${PROJECT_NAME}.7z" "${PROJECT_SOURCE_DIR}"
    WORKING_DIRECTORY
      "${PROJECT_BINARY_DIR}"
    )
else()
  message(
    WARNING "7-zip not found")
endif()
```