CMake Cheat Sheet

Get Latest Version:
https://thatonegamedev.com/

Command Line

Generate Project:

```
> cmake -S . -B ./build → Simple
> cmake -S . -B ./build -G Ninja → With Generator
> cmake -S . -B ./build --fresh → Reset Cache
> cmake -S . -B ./build --toolchain <path> → Cross-compile
> cmake -S . -B ./build --preset Dev → Use a preset from CMakePresets.json
```

Build Project:

```
> cmake --build ./build → Simple
> cmake --build ./build --config Release → Configuration
> cmake --build ./build --target Main → Only one target
> cmake --build ./build --parallel 8 → Build threads
```

Install Project:

```
> cmake --install ./build → Simple
> cmake --install ./build --config Release → Configuration
```

Variables

Setting a variable:

```
set(MY_VAR "VALUE")
set(MY_VAR VALUE)
set(MY_VAR VALUE PARENT_SCOPE) → When you want to have it
                                 outside of the current scope

set(MY_VAR VALUE CACHE STRING "Description") → Set & Cache
option(MY_VAR "Description" TRUE) → Cache for Booleans
```

Everything is a string:

```
set(MY_VAR "5") ⟷ set(MY_VAR 5)
set(MY_VAR 1 2 3) ⟷ set(MY_VAR "1;2;3")
```

Setting a variable from CLI:

```
> cmake -S . -B ./build -DMY_VAR=VALUE
```

Message

Most common message types

```
message(STATUS "Hello, CMake") → Prints always
message(DEBUG "Hello, CMake") → For CMake debugging
message(WARNING "Hello, CMake") → General warning
message(SEND_ERROR "Hello, CMake") → Error but process the rest
message(FATAL_ERROR "Hello, CMake") → Error and stop here
```

Variable Extraction

```
set(MY_VAR 42)
set(MY_OTHER "MY_VAR")

message(STATUS MY_VAR) → No extraction - "MY_VAR"
message(STATUS ${MY_VAR}) → Regular extraction - "42"
message(STATUS "${MY_VAR}3") → String extraction - "423"
message(STATUS ${${MY_OTHER}}) → Double extraction - "42"
```

List

```
set(LIST_VAR 1 2 3 3)

list(LENGTH LIST_VAR LENGTH_VAR) → 4
list(GET LIST_VAR 0 OUT_VAR) → 1
list(JOIN LIST_VAR ", " OUT_VAR) → "1, 2, 3, 3"
list(APPEND LIST_VAR 5) → "1;2;3;3;5"
list(INSERT LIST_VAR 3 4) → "1;2;3;4;3;5"
list(PREPEND LIST_VAR 0) → "0;1;2;3;4;3;5"
list(REMOVE_DUPLICATES LIST_VAR) → "0;1;2;3;4;5"
list(POP_BACK LIST_VAR) → "0;1;2;3;4"
list(POP_FRONT LIST_VAR) → "1;2;3;4"
list(REMOVE_ITEM LIST_VAR 4) → "1;2;3"
list(REMOVE_AT LIST_VAR 2) → "1;2"
list(REVERSE LIST_VAR) → "2;1"
list(SORT LIST_VAR) → "1;2"
```

String

```
string(TOLOWER "Hello" NEW_VAR) → "hello"
string(TOUPPER "Hello" NEW_VAR) → "HELLO"
string(LENGTH "Hello" NEW_VAR) → "5"
string(STRIP " Hello " NEW_VAR) → "Hello"
string(APPEND NEW_VAR "5") → "Hello5"
string(PREPEND NEW_VAR "5") → "5Hello5"
string(JOIN ", " NEW_VAR 1 2 3) → "1, 2, 3"
string(MD5 NEW_VAR "Hello") → Hashes the string
string(CONFIGURE "x = ${SOME_VAR}" NEW_VAR) → Uses CMake configure syntax
string(REPLACE " " ", " NEW_VAR "Hello World") → "Hello, World"

Also has REGEX_REPLACE
```

Booleans

True:

```
set(TRUTHY_VAR TRUE)
set(TRUTHY_VAR 1)
set(TRUTHY_VAR ON)
set(TRUTHY_VAR YES)
set(TRUTHY_VAR Y)
```

False:

```
set(FALSEY_VAR FALSE)
set(FALSEY_VAR 0)
set(FALSEY_VAR OFF)
set(FALSEY_VAR NO)
set(FALSEY_VAR N)
set(FALSEY_VAR IGNORE)
set(FALSEY_VAR NOTFOUND)
```

If & Else

Simple if:

```
if(<condition>)
    # Conditional code comes here
endif()
```

With an "Else":

```
if(<condition>)
    # Conditional code comes here
else()
    # Else code comes here
endif()
```

With an "Else If":

```
if(<condition>)
    # Conditional code comes here
elseif(<condition>)
    # Else if code comes here
else()
    # Else code comes here
endif()
```

THAT ONE GAME DEV

Number

```
math(EXPR NEW_INDEX_VAR "${INDEX} + 1")
```

Foreach

Foreah over list:

```
foreach(VALUE_VAR IN LISTS LIST_VAR)
    # Looped code comes here
endforeach()
```

Foreah over expanded list:

```
foreach(VALUE_VAR IN VALUES ${LIST_VAR})
    # Looped code comes here
endforeach()
```

Early Exit

In while and foreach:

```
break() → Exits loop
continue() → Moves to next iteration
```

While

Simple while:

```
while(<condition>)
    # Looped code comes here
endwhile()
```

Structure

```
include(Other.cmake) → Include CMake
add_subdirectory(Main) → Parse directory
```

Condition

```
if(VALUE1) ⟷ if(${VALUE1})
if(NOT VALUE1) → Inverse
if("ANYTHING") → Random values are FALSE

if(VALUE1 AND VALUE2) → True if both are TRUE
if(VALUE1 OR VALUE2) → True if any is TRUE
if((VALUE1 OR VALUE2) AND "TRUE") → Priority
```

```
if(COMMAND message) → True if command exists
if(TARGET Main) → True if target Main exists
if(DEFINED VALUE1) → True if variable exists

if(VALUE1 IN_LIST LIST_VAR) → True if list contains value
if(VALUE1 STREQUAL VALUE2) → True if variables are equal strings

if(VALUE1 PATHEQUAL VALUE2) → True if variables represent the same path
if(EXISTS "C:/file.txt") → True if path exists
if("C:/file1.txt" IS_NEWER_THAN "C:/file2.txt") → True if left file is newer
if(IS_DIRECTORY "C:/file.txt") → True if path represents a directory

if(VALUE1 LESS VALUE2) → True if left is a number lesser than right
if(VALUE1 LESS_EQUAL VALUE2) → True if left is a number lesser or equal than right
if(VALUE1 GREATER VALUE2) → True if left is a number greater than right
if(VALUE1 GREATER_EQUAL VALUE2) → True if left is a number greater or equal than right

The same exist for versions with VERSION_LESS, VERSION_GREATER, VERSION_EQUAL,
VERSION_LESSS_EQUAL, VERSION_GREATER_EQUAL
```

Files

```
file(READ "C:/file.txt" NEW_VAR) → Reads the file into NEW_VAR
file(TIMESTAMP "C:/file.txt" NEW_VAR) → Get last modified time
file(TOUCH "C:/file.txt") → Creates empty file if it doesn't exist
file(WRITE "C:/file.txt" "Hello") → Replaces file contents with "Hello"
file(APPEND "C:/file.txt" "Hello") → Adds "Hello" to end of file
file(CONFIGURE OUTPUT "C:/file.txt" CONTENT "x = ${SOME_VAR}") → Uses CMake configure syntax
file(GLOB SOURCES_VAR "*.cpp") → Gathers all .cpp files from the current directory
file(GLOB_RECURSE SOURCES_VAR "*.cpp") → Gathers all .cpp files from the current directory and child directories
file(MAKE_DIRECTORY "C:/directory") → Creates the directory
file(REMOVE "C:/directory") → Deletes file/directory
file(RENAME "C:/file.txt" "C:/other.txt") → Renames file.txt into other.txt
file(COPY "C:/file.txt" "C:/other.txt") → Copies file.txt into other.txt
file(DOWNLOAD "https://google.com/robots.txt" "C:/robots.txt") → Downloads file from the web
```

CMake Cheat Sheet

Targets

```
add_library(MyLibrary lib.h lib.cpp) → Simple library
add_library(MyLibrary INTERFACE lib.h) → Header only or facade for other libraries

add_executable(MyExe main.cpp) → Simple executable
add_executable(MyExe WIN32 main.cpp) → No terminal on windows

add_custom_target(MyCustom COMMANDS "python file.py") → Simple custom target
add_custom_target(MyCustom ALL COMMANDS "python file.py") → Build when all targets are built

add_dependencies(MyExe MyCustom) → MyExe will wait for MyCustom to build
target_link_libraries(MyExe PUBLIC MyLibrary) → MyExe will link to MyLibrary
```

Precompiler, Compiler & Linker

```
target_compile_definitions(MyExe PUBLIC FOO) → Adds a precomiler defintion
target_compile_definitions(MyExe PUBLIC FOO=1) → Adds a precompiler difinition with a value
target_include_directires(MyExe PUBLIC include/) → Adds an include directory

target_compile_options(MyExe PUBLIC /O2) → Adds a compiler flag
target_link_options(MyExe PUBLIC /OPT:REF) → Adds a linker flag
```

Common Variables

```
PROJECT_NAME → The name given to the last call to project()
PROJECT_IS_TOP_LEVEL → The current project is not a subdirectory
CMAKE_SOURCE_DIR → The location where cmake was called (root)
CMAKE_CURRENT_SOURCE_DIR → The location where the current CMakeLists.txt is (subdirectory)
CMAKE_BINARY_DIR → The location where cmake project is configured (root)
CMAKE_CURRENT_BINARY_DIR → The location where the current CMakeLists.txt is configured (subdirectory)
CMAKE_GENERATOR → The generator used for this configuration
CMAKE_TOOLCHAIN_FILE → The file used for cross-compilation

BUILD_SHARED_LIBS → Build with static or dynamic linking. Defaults to FALSE (static)
CMAKE_BUILD_TYPE → For single-configuration projects - Debug, Release, etc.
CMAKE_CONFIGURATION_TYPES → For multi-configuration projects - Debug, Release, etc.
CMAKE_MODULE_PATH → Path where CMake can find and use for its include() command
CMAKE_INSTALL_PREFIX → The path where you will install your project
CMAKE_MESSAGE_LOG_LEVEL → Sets log level for debugging

CMAKE_SYSTEM_NAME → Operating System name
CMAKE_SYSTEM_PROCESSOR → Processor type for which we are compiling
CMAKE_CXX_COMPILER_ID → The compiler being used
APPLE, ANDROID, BSD, IOS, LINUX, WIN32, WINCE, WINDOWS_PHONE, WINDOWS_STORE → Operating System shorts
BORLAND, CYGWIN, MSVC, MSYS, XCODE → Compiler environments

CMAKE_CXX_COMPILER_LAUNCHER → Set an application that will execute the compiler (like CCache)
CMAKE_CXX_CPPCHECK → Path to cppcheck and arguments (ninja and makefiles only)
CMAKE_CXX_CPPLINT → Path to cpplint and arguments (ninja and makefiles only)

CMAKE_COMMAND → Path to the cmake executable
CMAKE_CTEST_COMMAND → Path to the ctest executable
CMAKE_CPACK_COMMAND → Path to the cpack executable
CMAKE_MAKE_PROGRAM → Path to the program that builds the project (make, msbuild, etc.)
```

Functions & Macros

```
function(MyFunc BOUND_ARG)
    # Function code goes here (ARGV for all arguments, ARGN for not bound)
endfunction()

macro(MyMacro BOUND_ARG)
    # Function code goes here (ARGV for all arguments, ARGN for not bound)
endmacro()
```