3.7.3. Scripts

CMake can be used as a cross-platform scripting language.

cmake基本上就可以当成是一个跨平台的脚本语言来使用

CMake documentation

- CMake options

3.7.3.1. Example

Script for creating file:

```
# create-file.cmake

file(WRITE Hello.txt "Created by script")
```

Run script by `cmake -P`:

注意这里是 大写的 P

```
[cmake-sources]> rm -f Hello.txt
[cmake-sources]> cmake -P script/create-file.cmake
[cmake-sources]> ls Hello.txt
Hello.txt
[cmake-sources]> cat Hello.txt
Created by script
```

3.7.3.2. Minimum required (bad)

在脚本也使用cmake_minimum_required

在cmake脚本中也应该先使用这条命令

We should use `cmake_minimum_required` as the first command in script just like with the regular CMakeLists.txt. Lack of `cmake_minimum_required` may lead to problems:

```
# script.cmake

set("Jane Doe" "")
set(MYNAME "Jane Doe")

message("MYNAME: ${MYNAME}")

if("${MYNAME}" STREQUAL "")
  message("MYNAME is empty!")
endif()
```

```
[cmake-sources]> cmake -P minimum-required-bad/script.cmake
MYNAME: Jane Doe
CMake Warning (dev) at minimum-required-bad/script.cmake:6 (if):
  Policy CMP0054 is not set: Only interpret if() arguments as variables or
  keywords when unquoted.  Run "cmake --help-policy CMP0054" for policy
  details.  Use the cmake_policy command to set the policy and suppress this
  warning.

  Quoted variables like "Jane Doe" will no longer be dereferenced when the
  policy is set to NEW.  Since the policy is not set the OLD behavior will be
  used.
This warning is for project developers.  Use -Wno-dev to suppress it.

MYNAME is empty!
```

3.7.3.3. Minimum required (good)

Same example with `cmake_minimum_required` works correctly and without warning:

```
# script.cmake

cmake_minimum_required(VERSION 3.1)

set("Jane Doe" "")
set(MYNAME "Jane Doe")

message("MYNAME: ${MYNAME}")

if("${MYNAME}" STREQUAL "")
  message("MYNAME is empty!")
endif()
```

```
[cmake-sources]> cmake -P minimum-required-good/script.cmake
MYNAME: Jane Doe
```

3.7.3.4. cmake -E

-E cmake 命令行模式

使用cmake的删除命令就可以做到跨平台，因为不同平台的删除命令是不一样的

Example of using `cmake -E remove_directory` instead of native `rm` / `rmdir` commands:

CMake documentation

- Command-Line Tool Mode

```
cmake_minimum_required(VERSION 2.8)
project(foo NONE)

set(dir_to_remove "${CMAKE_CURRENT_BINARY_DIR}/__temp")

if(WIN32)
  # 'rmdir' will exit with error if directory doesn't exist
  # so we have to put 'if' here
  if(EXISTS "${dir_to_remove}")
    # need to convert to windows-style path
    file(TO_NATIVE_PATH "${dir_to_remove}" native_path)
    execute_process(
        COMMAND cmd /c rmdir "${native_path}" /S /Q
        RESULT_VARIABLE result
    )
  endif()
else()
  # no need to put 'if', 'rm -rf' produce no error if directory doesn't exist
  execute_process(
      COMMAND rm -rf "${dir_to_remove}"
      RESULT_VARIABLE result
  )
endif()

if(NOT result EQUAL 0)
  # Error
endif()
```

Same code with `cmake -E`:

```
cmake_minimum_required(VERSION 2.8)
project(foo NONE)
                    这个怎么确定          这个应该是cmake附带的一个命令
execute_process(
    COMMAND "${CMAKE_COMMAND}" -E remove_directory "${CMAKE_CURRENT_BINARY_DIR}/__temp"
    RESULT_VARIABLE result
)

if(NOT result EQUAL 0)
  # Error
endif()
```

Note

It's easier to use `file(REMOVE_RECURSE ...)` in this particular example