3.6.3. Environment variables

不是cmake中的环境变量，而是通过cmake来读取环境变量

cmake 与 系统环境变量之间的交互

3.6.3.1. Read

Environment variable can be read by using `$ENV{...}` syntax:

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

message("Environment variable USERNAME: $ENV{USERNAME}")
```

可以换成譬如 $ENV{PATH}

```
[usage-of-variables]> rm -rf _builds
[usage-of-variables]> echo $USERNAME
ruslo
[usage-of-variables]> export USERNAME
[usage-of-variables]> cmake -Hread-env -B_builds
Environment variable USERNAME: ruslo
-- Configuring done
-- Generating done
-- Build files have been written to: /.../usage-of-variables/_builds
```

3.6.3.2. Set

By using `set(ENV{...})` syntax CMake can set environment variable:

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

set(ENV{USERNAME} "Jane Doe")
message("Environment variable USERNAME: $ENV{USERNAME}")
```

```
[usage-of-variables]> rm -rf _builds
[usage-of-variables]> echo $USERNAME
ruslo
[usage-of-variables]> export USERNAME
[usage-of-variables]> cmake -Hset-env -B_builds
Environment variable USERNAME: Jane Doe
-- Configuring done
-- Generating done
-- Build files have been written to: /.../usage-of-variables/_builds
```

3.6.3.3. Unset

Unset environment variable:

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

unset(ENV{USERNAME})
message("Environment variable USERNAME: $ENV{USERNAME}")
```

```
[usage-of-variables]> rm -rf _builds
[usage-of-variables]> echo $USERNAME
ruslo
[usage-of-variables]> export USERNAME
[usage-of-variables]> cmake -Hunset-env -B_builds
Environment variable USERNAME:
-- Configuring done
-- Generating done
-- Build files have been written to: /.../usage-of-variables/_builds
```

3.6.3.4. Inheriting

Child process will inherit environment variables of parent:

```
# Top level CMakeLists.txt

cmake_minimum_required(VERSION 2.8)
project(foo NONE)

message("Set environment variable")

set(ENV{ABC} "This is ABC")

message("Top level ABC: $ENV{ABC}")

set(level1 "${CMAKE_CURRENT_LIST_DIR}/level1.cmake")

execute_process(
    COMMAND "${CMAKE_COMMAND}" -P "${level1}" RESULT_VARIABLE result
)

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

message("Unset environment variable")

unset(ENV{ABC})

message("Top level ABC: $ENV{ABC}")

execute_process(
    COMMAND "${CMAKE_COMMAND}" -P "${level1}" RESULT_VARIABLE result
)

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

```
# 'level1.cmake' script

message("Environment variable from level1: $ENV{ABC}")

set(level2 "${CMAKE_CURRENT_LIST_DIR}/level2.cmake")

execute_process(
    COMMAND "${CMAKE_COMMAND}" -P "${level2}" RESULT_VARIABLE result
)

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

```
# 'level2.cmake' script

message("Environment variable from level2: $ENV{ABC}")
```

```
[usage-of-variables]> rm -rf _builds
[usage-of-variables]> cmake -Henv-inherit -B_builds
Set environment variable
Top level ABC: This is ABC
Environment variable from level1: This is ABC
Environment variable from level2: This is ABC
Unset environment variable
Top level ABC:
Environment variable from level1:
Environment variable from level2:
-- Configuring done
-- Generating done
-- Build files have been written to: /.../usage-of-variables/_builds
```

3.6.3.5. Configure step

Note that in previous examples variable was set on configure step:

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

set(ENV{ABC} "123")

message("Environment variable ABC: $ENV{ABC}")

add_custom_target(
    foo
    ALL
    "${CMAKE_COMMAND}" -P "${CMAKE_CURRENT_LIST_DIR}/script.cmake"
)
```

```
[usage-of-variables]> rm -rf _builds
[usage-of-variables]> cmake -Henv-configure -B_builds
Environment variable ABC: 123
-- Configuring done
-- Generating done
-- Build files have been written to: /.../usage-of-variables/_builds
```

But environment variable remains the same on build step:

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

set(ENV{ABC} "123")

message("Environment variable ABC: $ENV{ABC}")

add_custom_target(
    foo
    ALL
    "${CMAKE_COMMAND}" -P "${CMAKE_CURRENT_LIST_DIR}/script.cmake"
)
```

```
# script.cmake

message("Environment variable from script: $ENV{ABC}")
```

```
[usage-of-variables]> cmake --build _builds
Scanning dependencies of target foo
Environment variable from script:
Built target foo
```

3.6.3.6. No tracking

CMake doesn't track changes of used environment variables so if your CMake code depends on environment variable and you're planning to change it from time to time it will break normal workflow:

```
cmake_minimum_required(VERSION 2.8)
project(foo)

set(target_name "$ENV{ABC}-tgt")
add_executable("${target_name}" foo.cpp)
```

Warning

Do not write code like that!

```
[usage-of-variables]> rm -rf _builds
[usage-of-variables]> export ABC=abc
[usage-of-variables]> cmake -Henv-depends -B_builds
-- The C compiler identification is GNU 4.8.4
-- The CXX compiler identification is GNU 4.8.4
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /.../usage-of-variables/_builds
[usage-of-variables]> cmake --build _builds
Scanning dependencies of target abc-tgt
[ 50%] Building CXX object CMakeFiles/abc-tgt.dir/foo.cpp.o
[100%] Linking CXX executable abc-tgt
[100%] Built target abc-tgt
```

Let's update environment variable:

```
[usage-of-variables]> export ABC=123
```

Name of the target **was not changed**:

```
[usage-of-variables]> cmake --build _builds
[100%] Built target abc-tgt
```

You have to run configure manually yourself:

```
[usage-of-variables]> cmake -Henv-depends -B_builds
-- Configuring done
-- Generating done
-- Build files have been written to: /.../usage-of-variables/_builds
[usage-of-variables]> cmake --build _builds
Scanning dependencies of target 123-tgt
[ 50%] Building CXX object CMakeFiles/123-tgt.dir/foo.cpp.o
[100%] Linking CXX executable 123-tgt
[100%] Built target 123-tgt
```

3.6.3.7. Summary

- CMake can set, unset and read environment variables
- Check carefully configure-build steps where you set environment variables
- Child processes will inherit environment variables of parent

- Do not make your CMake code depends on environment variable if that variable may change