

Snippets

Pieter P

CMake

Compilation timing

```
1 set_property(GLOBAL PROPERTY RULE_LAUNCH_COMPILE "${CMAKE_COMMAND} -E time")
```

Stripping debug information

```
1 # Strip and install debug information
2 function(myproject_install_debug_syms target component dest_lib dest_bin)
3     if (MSVC)
4         install(FILES "${CMAKE_PDB_FILE:${target}}"
5             DESTINATION ${dest_bin}
6             CONFIGURATIONS Debug RelWithDebInfo
7             COMPONENT ${component}
8             OPTIONAL EXCLUDE_FROM_ALL)
9     elseif (CMAKE_STRIP AND CMAKE_OBJCOPY)
10        set(DEBUG_FILE "${CMAKE_PDB_FILE:${target}}.debug")
11        add_custom_command(TARGET ${target} POST_BUILD
12            COMMAND "${CMAKE_STRIP}" "--only-keep-debug" "${CMAKE_PDB_FILE:${target}}" "-o" "${DEBUG_FILE}"
13            COMMAND "${CMAKE_STRIP}" "--strip-debug" "${CMAKE_PDB_FILE:${target}}"
14            COMMAND "${CMAKE_OBJCOPY}" "--add-gnu-debuglink=${DEBUG_FILE}" "${CMAKE_PDB_FILE:${target}}"
15            COMMAND "${CMAKE_COMMAND}" "-E" "echo" "Stripped into ${DEBUG_FILE}"
16            WORKING_DIRECTORY ${CMAKE_PDB_FILE_DIR:${target}})
17        install(FILES "${CMAKE_PDB_FILE_DIR:${target}}/${DEBUG_FILE}"
18            DESTINATION ${dest_lib}
19            CONFIGURATIONS Debug RelWithDebInfo
20            COMPONENT ${component}
21            EXCLUDE_FROM_ALL)
22    endif()
23 endfunction()
24
25 # Usage
26 include(GNUInstallDirs)
27 foreach(target IN LISTS MYPROJECT_INSTALL_TARGETS)
28     get_target_property(target_type ${target} TYPE)
29     if (${target_type} STREQUAL "SHARED_LIBRARY")
30         myproject_install_debug_syms(${target} debug
31             ${CMAKE_INSTALL_LIBDIR}
32             ${CMAKE_INSTALL_BINDIR})
33     endif()
34 endforeach()
```

Hiding symbols with default visibility in shared library

```
1 function(configure_visibility target)
2     set_target_properties(${target} PROPERTIES CXX_VISIBILITY_PRESET "hidden"
3         VISIBILITY_INLINES_HIDDEN true)
4     if (CMAKE_SYSTEM_NAME MATCHES "Linux")
5         target_link_options(${target} PRIVATE "LINKER:--exclude-libs,ALL")
6     endif()
7 endfunction()
```

The target properties ensure that the source files comprising the given target are compiled with the visibility set to hidden, which means that unless a symbol is explicitly marked “export”, it won’t be exported in the shared library.

The linker option ensures that symbols with default visibility in any static libraries the target depends on are not exported in the shared library either.

From `man ld(1)`:

```
--exclude-libs lib,lib,...
Specifies a list of archive libraries from which symbols should not
be automatically exported. The library names may be delimited by
commas or colons. Specifying "--exclude-libs ALL" excludes symbols
in all archive libraries from automatic export.
```

For explicitly exporting symbols that are part of the public API, see [GenerateExportHeader](#).

Inspecting binaries

```
1 # Dependencies and dynamic section of a shared library
2 readelf -d libfile.so
3 # List of symbols in shared library (1)
4 nm -CD --defined-only --size-sort libfile.so
5 # List of symbols in shared library (2)
6 readelf --wide --symbols --demangle libfile.so
```

C++

Reversing a linked list

```
1 #include <utility> // std::exchange
2
3 struct Node {
4     Node *next = nullptr;
5 };
6
7 Node *reverse_linked_list(Node *fwd) {
8     Node *rev = nullptr;
9     while (fwd)
10         rev = std::exchange(fwd, std::exchange(fwd->next, rev));
11     return rev;
12 }
```
