安装第三方库

- English
- 我的 VSCode C++ 配置教程中对此的图文介绍

本模板提供了两种安装第三方库的方式. 你可以混用它们来安装库.

⚠️ 请确保你的网络——尤其是**终端 (terminal)** 的网络——可以访问 github, 例如 "steam++工具箱" 中可以选择加速 github 访问. 如果使用科学工具, 请开启 "tun mode" 或 "增强模式" 之类的选项.

使用 vcpkg

开启 vcpkg

编辑 `CMakeLists.txt` (位于本仓库文件夹的根目录下), 在 `include(cpp_novice_fetch_project_options)` 和 `project(cpp_novice LANGUAGES CXX)` 之间添加一行 `run_vcpkg()`. 即:

```cmake
cmake_minimum_required(VERSION 3.25)

list(APPEND CMAKE_MODULE_PATH "${CMAKE_CURRENT_SOURCE_DIR}/cmake")
include(fix_msvc)
include(enable_cpp_module NO_POLICY_SCOPE)
include(cpp_novice_fetch_project_options)

run_vcpkg()
project(cpp_novice LANGUAGES CXX)
```

通过添加这一行代码, 我们启用 cmake 自动安装和使用 vcpkg, 因此我们之后只需要*指明需要的库*和*将库与程序链接*.

搜索库

首先, 我们打开 Browse public vcpkg packages 来搜索需要的库, 从而了解它们是否存在以及确切的库名是什么.

接下来我会以 `fmt` 和 `range-v3` 库为例.

指明需要的库

编辑 `vcpkg.json` (位于本仓库文件夹的根目录下), 将库名加入到 `dependencies` 数组中.

例如, 以下是我们如何添加 `fmt` 和 `range-v3` 库:

```json
{
  "$schema": "https://raw.githubusercontent.com/microsoft/vcpkg-tool/main/docs/vcpkg.schema.json",
  "dependencies": ["fmt", "range-v3"]
}
```

以某种方式清除 cmake 缓存并重新配置你 IDE 中的 cmake (例如, 你也许可以删除 build 或 out 文件夹, 并重启软件). 如果运气好的话, cmake 将会为你自动安装 vcpkg, 并使用它安装你在 `dependencies` 中指明的库.

将库与程序链接

在重启你的 IDE (或重新配置 cmake) 之后, cmake 应该在某处输出类似于以下例子的消息, 来告知你如何使用安装的库:

```
[cmake] The package fmt provides CMake targets:
[cmake]
[cmake]     find_package(fmt CONFIG REQUIRED)
[cmake]     target_link_libraries(main PRIVATE fmt::fmt)
[cmake]
[cmake]     # Or use the header-only version
[cmake]     find_package(fmt CONFIG REQUIRED)
[cmake]     target_link_libraries(main PRIVATE fmt::fmt-header-only)
[cmake]
[cmake] range-v3 provides CMake targets:
[cmake]
[cmake]   # this is heuristically generated, and may not be correct
[cmake]   find_package(range-v3 CONFIG REQUIRED)
[cmake]   target_link_libraries(main PRIVATE range-v3::meta range-v3::concepts range-v3::range-v3)
[cmake]
[cmake] -- Running vcpkg install - done
```

虽然这个指示已经足够简单, 我在这个模板中对它进行了进一步简化. 你只需要编辑 `CMakeLists.txt` 中的 `add_program_options`:

- 添加在 `find_package(<name> CONFIG REQUIRED)` 给出的库名到 `DEPENDENCIES` 中.
- 添加在 `target_link_libraries(main PRIVATE <name>)` 给出的库名到 `LIBRARIES` 中.

例如, 以下是我们如何添加 `fmt` 和 `range-v3` 库:

```
add_program_options(
  DEPENDENCIES
  fmt
  range-v3

  LIBRARIES
  fmt::fmt
  range-v3::meta
  range-v3::concepts
  range-v3::range-v3

  INCLUDES
  include
)
```

再一次重启你的 IDE (或以某种方式重新配置你 IDE 中的 cmake).

安装完成.

使用 conan

以某种方式安装 conan. 例如, 你可以从官网下载它, 或参考 我的 VSCode C++ 配置教程.

开启 conan

编辑 `CMakeLists.txt` (位于本仓库文件夹的根目录下), 在 `include(cpp_novice_fetch_project_options)` 和 `project(cpp_novice LANGUAGES CXX)` 之间添加一行 `run_conan()`. 即:

```
cmake_minimum_required(VERSION 3.25)

list(APPEND CMAKE_MODULE_PATH "${CMAKE_CURRENT_SOURCE_DIR}/cmake")
include(fix_msvc)
include(enable_cpp_module NO_POLICY_SCOPE)
include(cpp_novice_fetch_project_options)

run_conan()
project(cpp_novice LANGUAGES CXX)
```

通过添加这一行代码, 我们启用 cmake 使用 conan, 因此我们之后只需要*指明需要的库*和*将库与程序链接*.

搜索库

首先, 我们打开 JFrog ConanCenter 来搜索需要的库, 从而了解它们是否存在、确切的库名是什么和最新的版本号.

接下来我会以 `fmt` 和 `range-v3` 库为例.

指明需要的库

编辑 `conanfile.txt` (位于本仓库文件夹的根目录下), 将 `<library_name>/<version>` 加入到 `[requires]` 下.

例如, 以下是我们如何添加 `fmt/10.2.1` 和 `range-v3/0.12.0` 库:

```
[layout]
cmake_layout

[requires]
fmt/10.2.1
range-v3/0.12.0

[generators]
CMakeDeps
```

以某种方式清除 cmake 缓存并重新配置你 IDE 中的 cmake (例如, 你也许可以删除 build 或 out 文件夹, 并重启软件). 如果运气好的话, cmake 将自动使用 conan 安装你在 `[requires]` 中指明的库.

将库与程序链接

在重启你的 IDE (或重新配置 cmake) 之后, cmake 应该在某处输出类似于以下例子的消息, 来告知你如何使用安装的库:

```
[cmake] conanfile.txt: CMakeDeps necessary find_package() and targets for your CMakeLists.txt
[cmake]     find_package(fmt)
[cmake]     find_package(range-v3)
[cmake]     target_link_libraries(... fmt::fmt range-v3::range-v3)
[cmake] conanfile.txt: Generating aggregated env files
[cmake] conanfile.txt: Generated aggregated env files: ['conanbuild.sh', 'conanrun.sh']
[cmake] Install finished successfully
```

虽然这个指示已经足够简单, 我在这个模板中对它进行了进一步简化. 你只需要编辑 `CMakeLists.txt` 中的 `add_program_options`:

- 添加在 `find_package(<name>)` 给出的库名到 `DEPENDENCIES` 中.
- 添加在 `target_link_libraries(... <name>)` 给出的库名到 `LIBRARIES` 中.

例如, 以下是我们如何添加 `fmt` 和 `range-v3` 库:

```
add_program_options(
  DEPENDENCIES
  fmt
  range-v3

  LIBRARIES
  fmt::fmt
  range-v3::range-v3

  INCLUDES
  include
)
```

再一次重启你的 IDE (或以某种方式重新配置你 IDE 中的 cmake).

安装完成.

使用 CMake 自带的某些第三方库支持

⚠️ 建议通过《Modern CMake for C++》学习 CMake.

前面提到的 vcpkg 和 conan, 都是为你下载、(编译)、安装第三方库后, 再向负责 C++ 项目管理的 CMake 提供 `Find<PackageName>.cmake`、`<package_name>-config.cmake` 或 `<PackageName>Config.cmake` 文件, 从而让 CMake 能成功找到并使用这些第三方库.

CMake 本身也提供了一些第三方库的支持 (如 `FindOpenMP.cmake`, `FindQt.cmake` 等), 因此只要你安装这些库到系统路径后, 即可在 CMake 中使用:

```
add_program_options(
  DEPENDENCIES
  OpenMP

  LIBRARIES
  OpenMP::OpenMP_CXX

  INCLUDES
  include
)
```

以某种方式清除 cmake 缓存并重新配置你 IDE 中的 cmake (例如, 你也许可以删除 build 或 out 文件夹, 并重启软件).

安装完成.