Install third-party libraries

中文

The template provides two ways of third-party library installation. You can mix them to install libraries if you like.

⚠️ Please make sure the network is accessible to github.

Use vcpkg

Enable vcpkg

Edit `CMakeLists.txt` (located at the root of this repository folder), add a line `run_vcpkg()` between `include(cpp_novice_fetch_project_options)` and `project(cpp_novice LANGUAGES CXX)`. That is:

```
cmake_minimum_required(VERSION 3.25)

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

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

By adding this line of code, we enable cmake to install and use vckpg automatically, so all we need to do next is to specify libraries required and link libraries to our programs.

Search libraries

First, we open the website Browse public vcpkg packages to search libraries we need in order to see whether them exist and what their exact names are.

Below I'll use `fmt` and `range-v3` libraries as the example.

Specify libraries required

Edit `vcpkg.json` (located at the root of this repository file) by simply adding the library names into the `dependencies` array.

For example, here's how we add `fmt` and `range-v3` library:

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

Reopen your IDE (or reconfigure cmake in your IDE somehow). If you're lucky, cmake will automatically install vcpkg for you, and use it to install `dependencies` you specified.

Link libraries to our programs

After reopening your IDE (or reconfigure cmake), cmake should output messages somewhere like the following example, which inform you on how to use the installed libraries:

```
[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
```

Although the instruction is already simple, I simplified this in the template even further. Just edit the `add_program_options` function in `CMakeLists.txt` by:

- adding library names showed in `find_package(<name> CONFIG REQUIRED)` into `DEPENDENCIES` section.
- adding target names showed in `target_link_libraries(main PRIVATE <name>)` into `LIBRARIES` section.

For example, here's how we add `fmt` and `range-v3` library:

```
add_program_options(
  DEPENDENCIES
  fmt
  range-v3

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

  INCLUDES
  include
)
```

Reopen your IDE (or reconfigure cmake in your IDE somehow) again.

Done.

Use conan

Install conan

Install conan somehow. For example, you can download it from the official website.

Enable conan

Edit `CMakeLists.txt` (located at the root of this repository file), add a line `run_conan()` between `include(cpp_novice_fetch_project_options)` and `project(cpp_novice LANGUAGES CXX)`. That is:

```cmake
cmake_minimum_required(VERSION 3.25)

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

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

By adding this line of code, we enable cmake to install and use vckpg automatically, so all we need to do next is to specify libraries required and link libraries to our programs.

Search libraries

First, we open the website JFrog ConanCenter to search libraries we need in order to see whether they exist, what their exact names are and what their latest version are.

Below I'll use `fmt/10.2.1` and `range-v3/0.12.0` libraries as the example.

Specify libraries required

Edit `conanfile.txt` (located at the root of this repository file) by simply adding the `<library_name>/<version>` into the `[requires]` section.

For example, here's how we add `fmt/10.2.1` and `range-v3/0.12.0` library:

```
[layout]
cmake_layout

[requires]
fmt/10.2.1
range-v3/0.12.0

[generators]
CMakeDeps
```

Reopen your IDE (or reconfigure cmake in your IDE somehow). If you're lucky, cmake will automatically use conan to install `[requires]` you specified.

Link libraries to our programs

After reopening your IDE (or reconfigure cmake), cmake should output messages somewhere like the following example, which inform you on how to use the installed libraries:

```
[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
```

Although the instruction is already simple, I simplified this in the template even further. Just edit the `add_program_options` function in `CMakeLists.txt` by:

- adding library names showed in `find_package(<name>)` into `DEPENDENCIES` section.
- adding target names showed in `target_link_libraries(... <name>)` into `LIBRARIES` section.

For example, here's how we add `fmt` and `range-v3` library:

```
add_program_options(
  DEPENDENCIES
  fmt
  range-v3

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

  INCLUDES
  include
)
```

Reopen your IDE (or reconfigure cmake in your IDE somehow) again.

Done.