Managing Libraries

2024.04.04

SWPP Practice Session

Seunghyeon Nam

Compiling with Dependency

- Many projects are dependent to some libraries
  - Standard C++ library (libc++, libstdc++, ...)
  - LLVM library
- We need two things to use library in the project
  - Interface to access the library functions
  - Binary that contains the actual implementation

Compiling with Dependency

- Interface: Header or module
  - Declarations should be included/imported into the code
  - Code will not compile without include/import
- Binary: shared or static library
  - Declarations should be linked to the actual definition
  - Linker will fail if declarations are not matched with implementation

Library Convention

- Libraries are installed into `/usr` or `/usr/local` by default
  - These are ‘system library path’
  - Headers reside in `.../include`
  - Libraries reside in `.../lib`
- Package managers install libraries in `/usr`
- User-built libraries are installed in `/usr/local`

Library Convention

- Linkers look for libraries in the system library path by default
  - You don't have to add the library path to the project by yourself
- You cannot keep multiple versions of the same library
  - Installing newer version will overwrite the older one!

Library Convention

- You may install the library in a discrete location
  - Headers reside in `<library>/include`
  - Libraries reside in `<library>/lib`
- Both include directory and library directory must be known to the project to use the library

Library Convention

- Installing library in discrete location has many benefits
  - You can maintain multiple versions of the same library
  - You don’t need root privilege to install a new library
  - You can remove the entire library simply by removing the directory

- Prefer installing each library in discrete location!

Linker Search Path

- By default, linkers only search in system library path
  - Standard C++ library seems like an exception
  - When the compiler calls the linker, it adds the library path as well
- Other libraries should be added to the linker search path
- In CMake, use `link_directories()`

Runtime Linker Search Path

- When a program is linked with dynamic library, these libraries should be linked at runtime.
- Linux and macOS has a separate ‘runtime linker’
  - In Linux, it refers to the dynamic library cache
  - In macOS, it refers to the library path embedded in the binary

Runtime Linker Search Path

- In Linux, you can manually add another search path
- Write a `<config-file-name>.conf` in `/etc/ld.so.conf.d`
  - Add only one path in each .conf file!
  - You need sudo to create or edit file in `/etc/ld.so.conf.d`
- Then, update the dynamic library cache
  - `sudo ldconfig`

Runtime Linker Search Path

- TL;DR : Add these search paths if you're using Linux
  - `<llvm-install-dir>/lib` (LLVM)
  - `<llvm-install-dir>/lib/x86_64-unknown-linux-gnu` (libc++)