Working with C++ packages with Vcpkg on Ubuntu environment

Park DongHa
github.com/luncliff, C++Korea

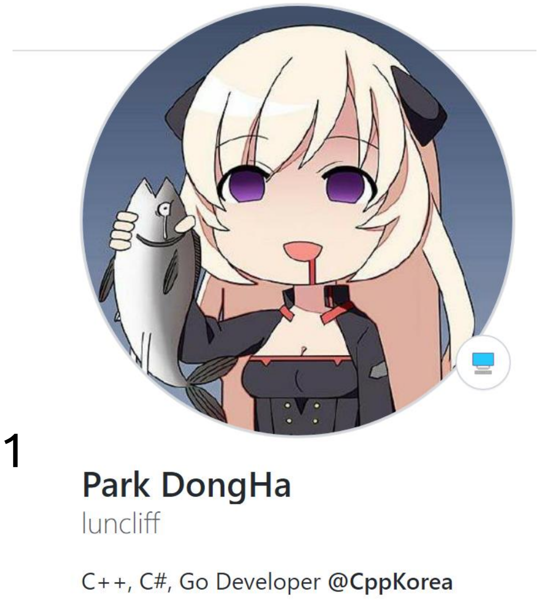
About The Speaker

Interests?

- C++, Programming Language Implementations
- Buildsystem integration of Open-Source projects
- Managing Software Capitals in organization level

Using Vcpkg since 2018. Sometimes contribute since 2021

Mostly develop on Windows & WSL.
Couldn't work on Ubuntu for years… 😢

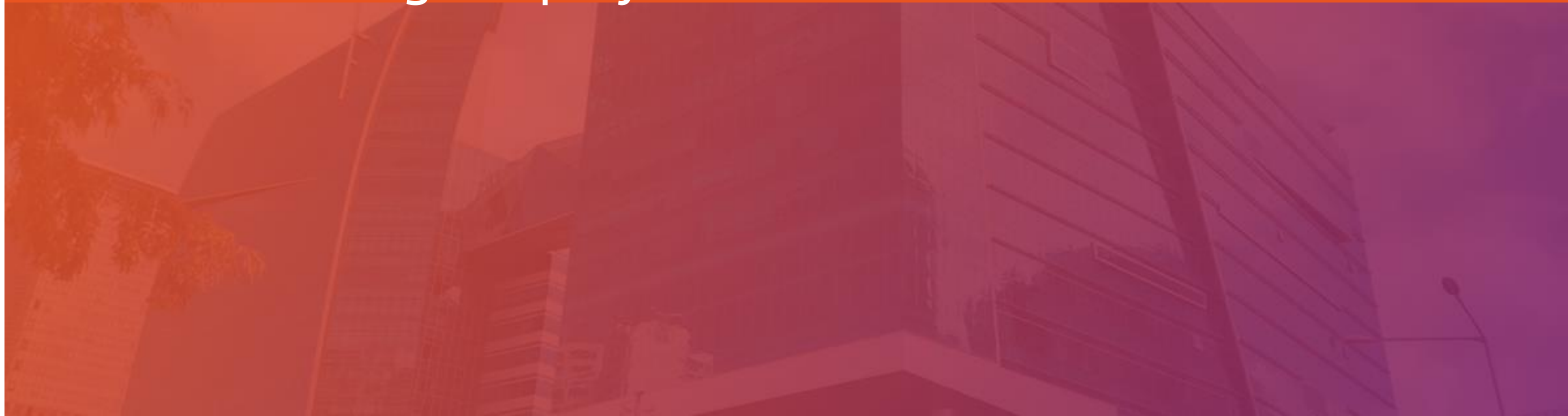

Park DongHa
luncliff
C++, C#, Go Developer **@CppKorea**

About The Workshop

- 3 Parts
  - Presentation Slides
  - Short demo, Explanations
  - Question times
- Wrap-up in the end
  - Summary
  - More demo, question time

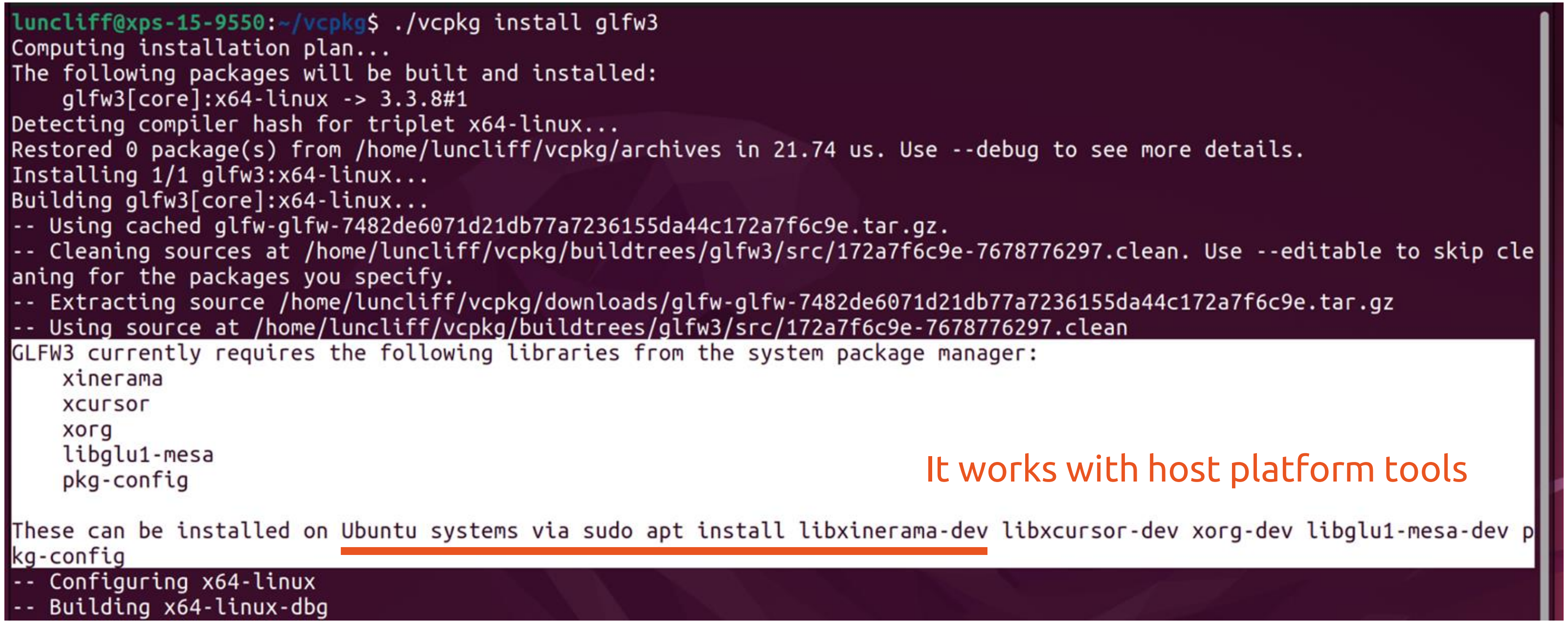
Part 1: Vcpkg

Vcpkg characteristics
How it solves library management problems
Understanding the project’s files



The Topic - Vcpkg

Package manager which installs C/C++ **Libraries**.

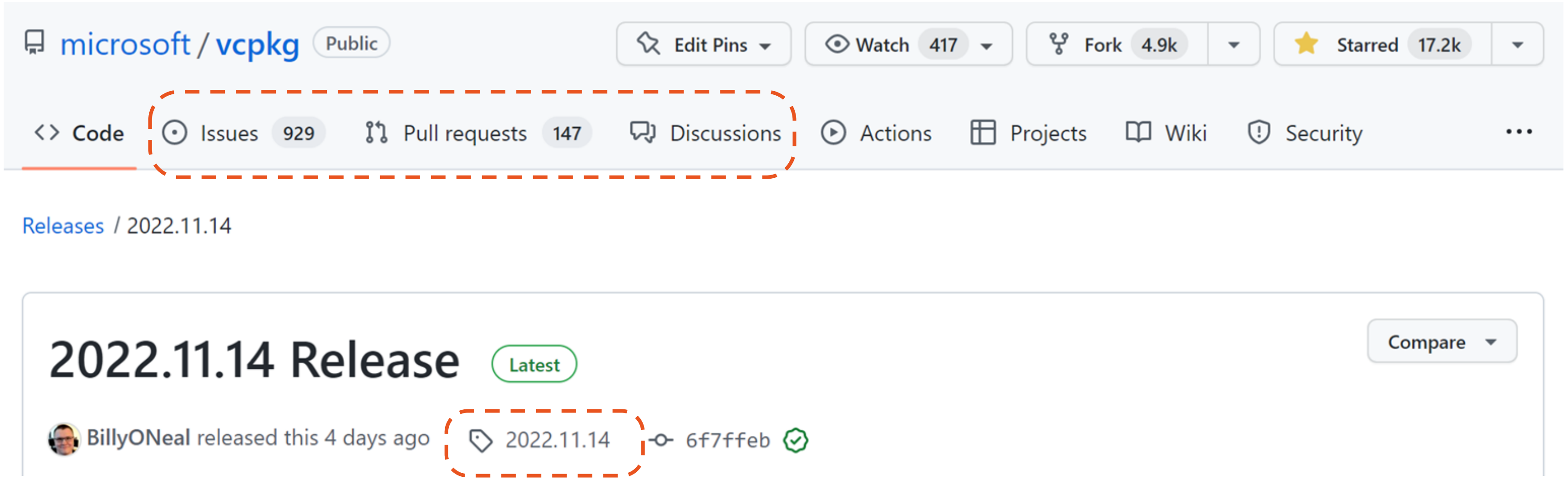


```
luncliff@xps-15-9550:~/vcpkg$ ./vcpkg install glfw3
Computing installation plan...
The following packages will be built and installed:
    glfw3[core]:x64-linux -> 3.3.8#1
Detecting compiler hash for triplet x64-linux...
Restored 0 package(s) from /home/luncliff/vcpkg/archives in 21.74 us. Use --debug to see more details.
Installing 1/1 glfw3:x64-linux...
Building glfw3[core]:x64-linux...
-- Using cached glfw-glfw-7482de6071d21db77a7236155da44c172a7f6c9e.tar.gz.
-- Cleaning sources at /home/luncliff/vcpkg/buildtrees/glfw3/src/172a7f6c9e-7678776297.clean. Use --editable to skip cle
aning for the packages you specify.
-- Extracting source /home/luncliff/vcpkg/downloads/glfw-glfw-7482de6071d21db77a7236155da44c172a7f6c9e.tar.gz
-- Using source at /home/luncliff/vcpkg/buildtrees/glfw3/src/172a7f6c9e-7678776297.clean
GLFW3 currently requires the following libraries from the system package manager:
    xinerama
    xcursor
    xorg
    libglu1-mesa
    pkg-config

These can be installed on Ubuntu systems via sudo apt install libxinerama-dev libxcursor-dev xorg-dev libglu1-mesa-dev p
kg-config
-- Configuring x64-linux
-- Building x64-linux-dbg
```

It works with host platform tools

Where Can I Find Vcpkg?

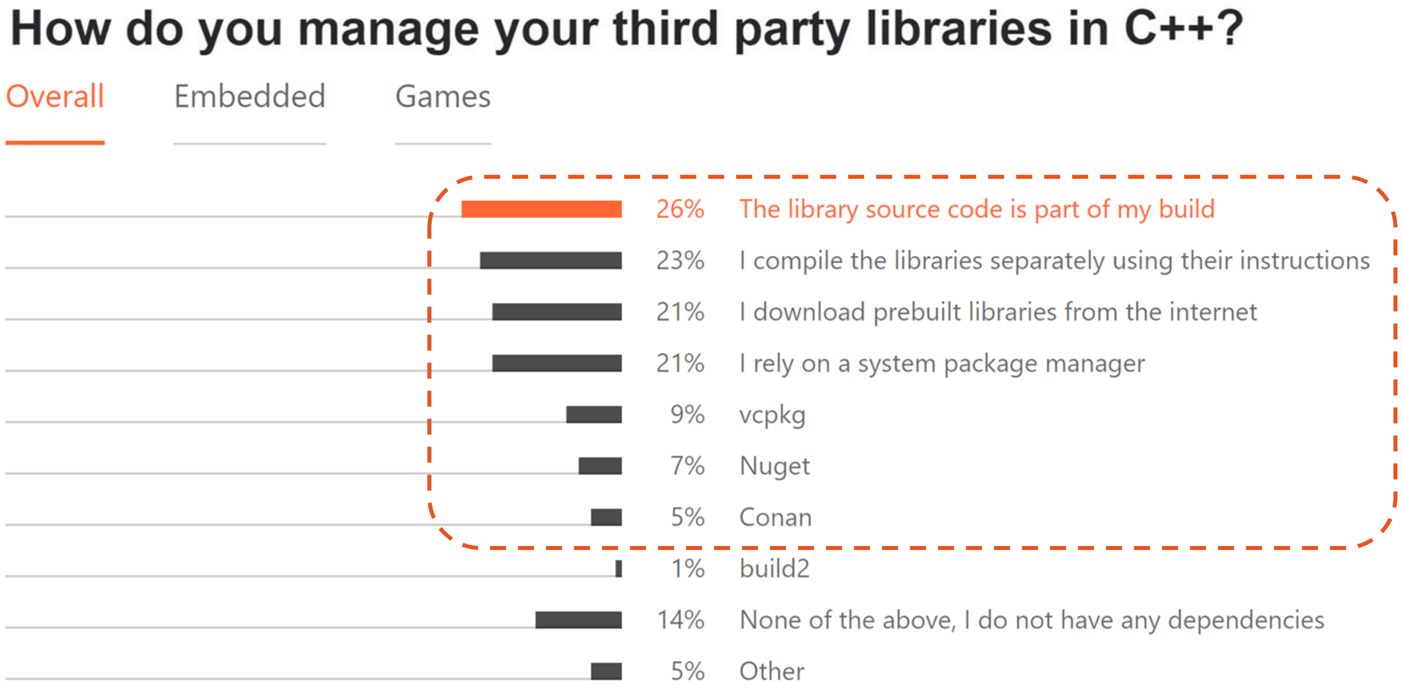
GitHub: https://github.com/microsoft/vcpkg



Visited 2022/11/18

Is Vcpkg That Famous?

Not Yet.



https://www.jetbrains.com/lp/devecosystem-2021/cpp/

More Details?

Vcpkg Official Documentation

https://vcpkg.io/en/docs/README.html

CppCon 2022 (YouTube @CppCon)

“C++ Dependencies Don't Have To Be Painful!” - Augustin Popa

Microsoft C++ Team Blog

https://devblogs.microsoft.com/cppblog/

What Are Its Characteristics?

Build programs from sources
(Even for host platform executables)

Works in 1 folder (We call it Vcpkg Root)

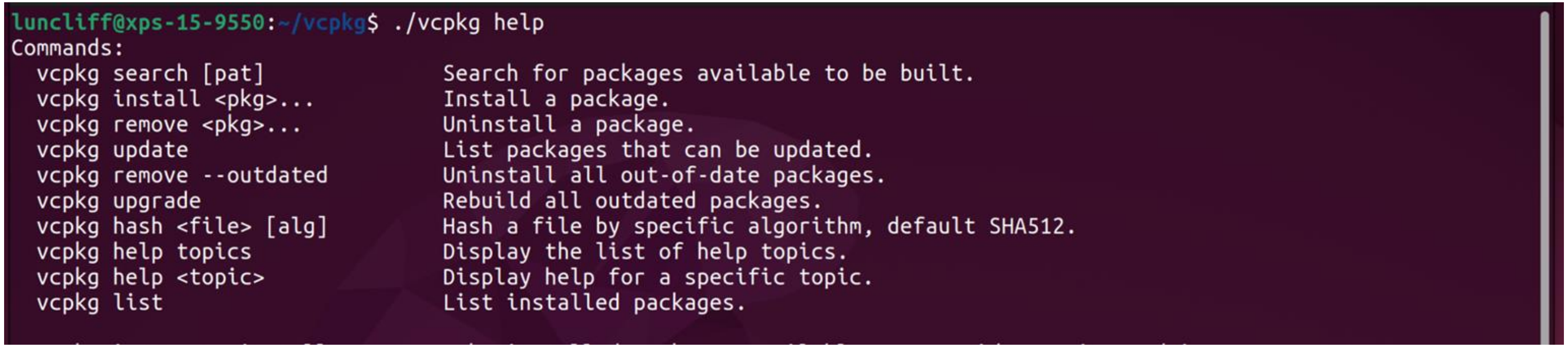
- Easy to search script, descriptions, logs, etc.
- Easy to cleanup after install

Environment variables can affect the behavior

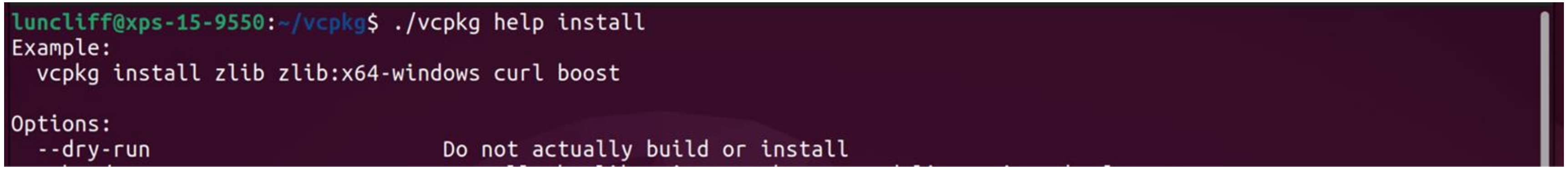
Download, caching for **reproducible developer environment**

Vcpkg Commands - 1

After bootstrap…



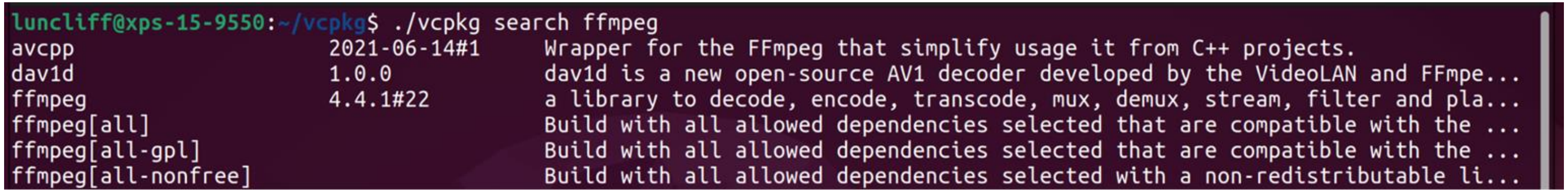
```
luncliff@xps-15-9550:~/vcpkg$ ./vcpkg help
Commands:
  vcpkg search [pat]              Search for packages available to be built.
  vcpkg install <pkg>...          Install a package.
  vcpkg remove <pkg>...           Uninstall a package.
  vcpkg update                    List packages that can be updated.
  vcpkg remove --outdated         Uninstall all out-of-date packages.
  vcpkg upgrade                   Rebuild all outdated packages.
  vcpkg hash <file> [alg]         Hash a file by specific algorithm, default SHA512.
  vcpkg help topics               Display the list of help topics.
  vcpkg help <topic>              Display help for a specific topic.
  vcpkg list                      List installed packages.
```



```
luncliff@xps-15-9550:~/vcpkg$ ./vcpkg help install
Example:
  vcpkg install zlib zlib:x64-windows curl boost

Options:
  --dry-run                       Do not actually build or install
```

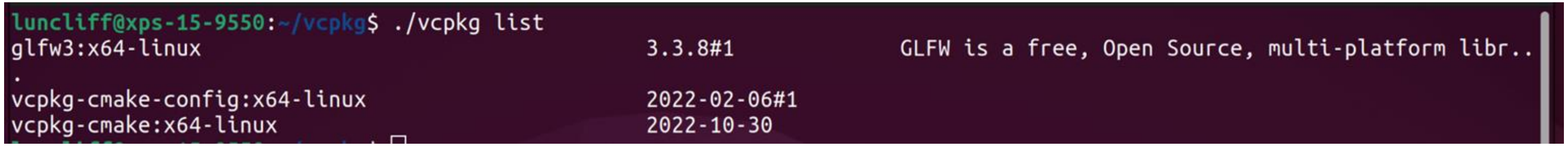
Vcpkg Commands - 2

Search available libraries …



```
luncliff@xps-15-9550:~/vcpkg$ ./vcpkg search ffmpeg
avcpp                    2021-06-14#1     Wrapper for the FFmpeg that simplify usage it from C++ projects.
dav1d                    1.0.0            dav1d is a new open-source AV1 decoder developed by the VideoLAN and FFmpe...
ffmpeg                   4.4.1#22         a library to decode, encode, transcode, mux, demux, stream, filter and pla...
ffmpeg[all]                               Build with all allowed dependencies selected that are compatible with the ...
ffmpeg[all-gpl]                           Build with all allowed dependencies selected that are compatible with the ...
ffmpeg[all-nonfree]                       Build with all allowed dependencies selected with a non-redistributable li...
```

And list installed libraries …



```
luncliff@xps-15-9550:~/vcpkg$ ./vcpkg list
glfw3:x64-linux                         3.3.8#1            GLFW is a free, Open Source, multi-platform libr..
.
vcpkg-cmake-config:x64-linux            2022-02-06#1
vcpkg-cmake:x64-linux                   2022-10-30
```

The Problem: Library Management

A software is implemented based on other softwares

- Knowledge stress to understand the “dependencies”
  - Requirements for each of softwares
  - How to install, use, and deploy them
- More efforts to be “cooperative“
  - Continuous bug reporting
  - Compatibility

More dependencies = More Stress

Higher the level, harder to manage. (Personal → Organization → Enterprise)

How Can We Solve It?

Package Manager. It helps you to solve some complicated scenarios.

- Easy commands
- Warning/Error messages
- Predictable rules

Every developers have their own style, but a package manager doesn't.

→ We can share the style by using the same tool

How Vcpkg Manages The Libraries?

Diagnosable Install

- Build from sources (logs, option can be diagnosed)
- Dependent packages after already validated

Consistent Pattern

- GNU style (${prefix}/include, ${prefix}/lib, etc)

Deploy

- Support zip creation

Understanding The Project's File Structure

After git clone?

- docs/
- scripts/
  - bootstrap-vcpkg.sh will download executable from github.com/microsoft/vcpkg-tool
- triplets/
- ports/

Triplets and Ports are written with CMake scripts. (We will see soon)

Basic level users won't need to care the other folders.

Understanding The Project’s File Structure

After port installation

- buildtrees/
  - Intermediate build files, logs
- downloads/
  - Download sources, patches, resource files
- packages/
  - Staging folder before copying to `installed/`
- installed/
  - Aggregation of installed packages

Vcpkg Files - Triplets

Triplet files are used to provide 2 things

1. Target Environment
2. Overall configurations for ports to install

```cmake
# triplets/x64-linux.cmake
set(VCPKG_TARGET_ARCHITECTURE x64)
set(VCPKG_CRT_LINKAGE dynamic)
set(VCPKG_LIBRARY_LINKAGE static)

set(VCPKG_CMAKE_SYSTEM_NAME Linux)
```

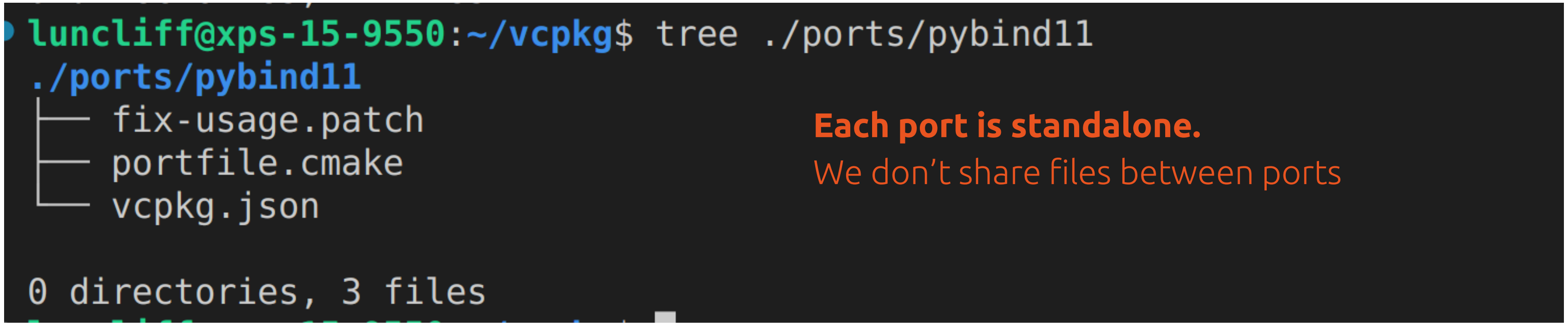
← Targeting AMD64 arch

← If not specified, output libs will be **.a** file

Vcpkg Files - Ports

A port is an implementation of package installation.

- **portfile.cmake** - where to download source files, installation of package
- **vcpkg.json** - name, version, description, dependencies
- **patch files** - Sometimes we need edits to integrate with other ports



```
luncliff@xps-15-9550:~/vcpkg$ tree ./ports/pybind11
./ports/pybind11
├── fix-usage.patch
├── portfile.cmake
└── vcpkg.json

0 directories, 3 files
```

Each port is standalone.
We don't share files between ports

Demo 1 - Understanding Files of Vcpkg

There will be a question time after this!

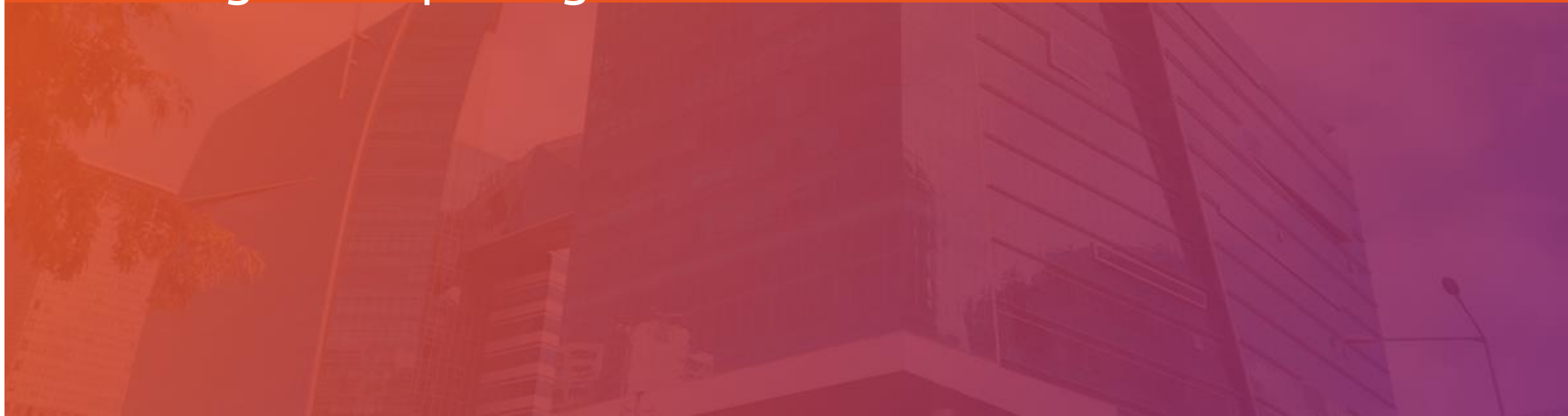
1. Setup / Cleanup / Basic Commands
   - git clone and bootstrap
   - CLI output of the commands (What can we see if --help?)
   - File structure explanation
2. Triplets
   - Compare existing triplets
   - static / dynamic
3. Ports
   - Files in the port
   - General steps in the portfile.cmake

Part 2: Scripting in Vcpkg

Basic CMake syntax for scripting

Importing existing packages

Writing a new package



CMake - Cross-platform Make

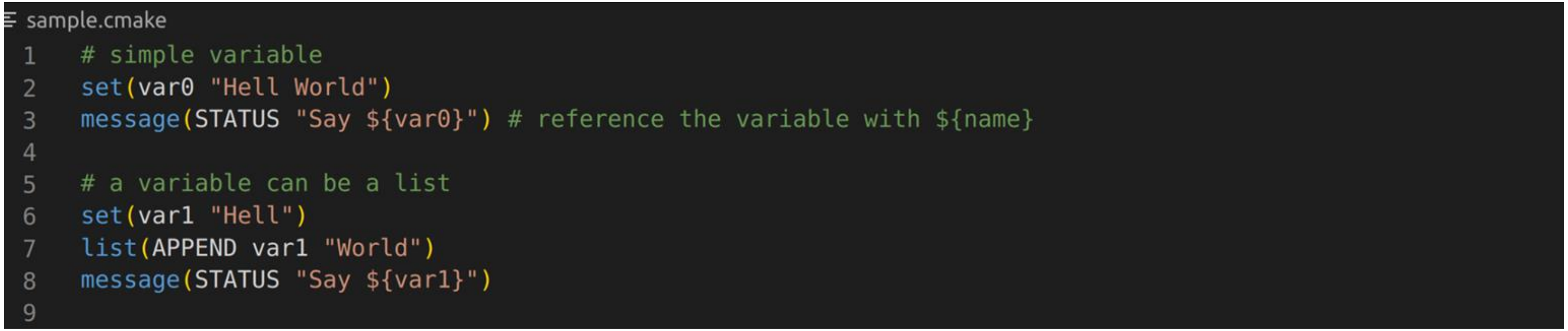
CMake is a build-system **generator**.
Generate **Unix Makefiles**, **Ninja** files from CMake files.

Official Tutorial
https://cmake.org/cmake/help/latest/guide/tutorial/index.html

Note: There are much of old materials.
If you want/have to learn, search with "Modern CMake"

CMake - Basic Commands

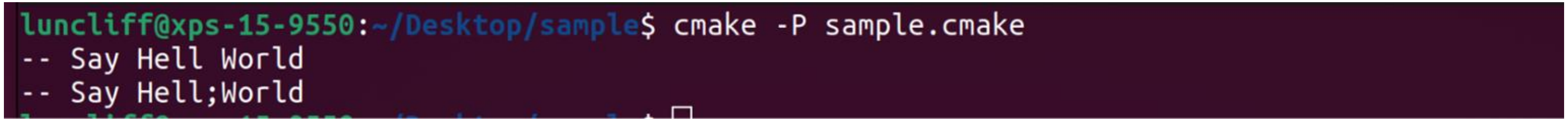
Make 2 variables and print it



sample.cmake

```cmake
# simple variable
set(var0 "Hell World")
message(STATUS "Say ${var0}") # reference the variable with ${name}

# a variable can be a list
set(var1 "Hell")
list(APPEND var1 "World")
message(STATUS "Say ${var1}")

```

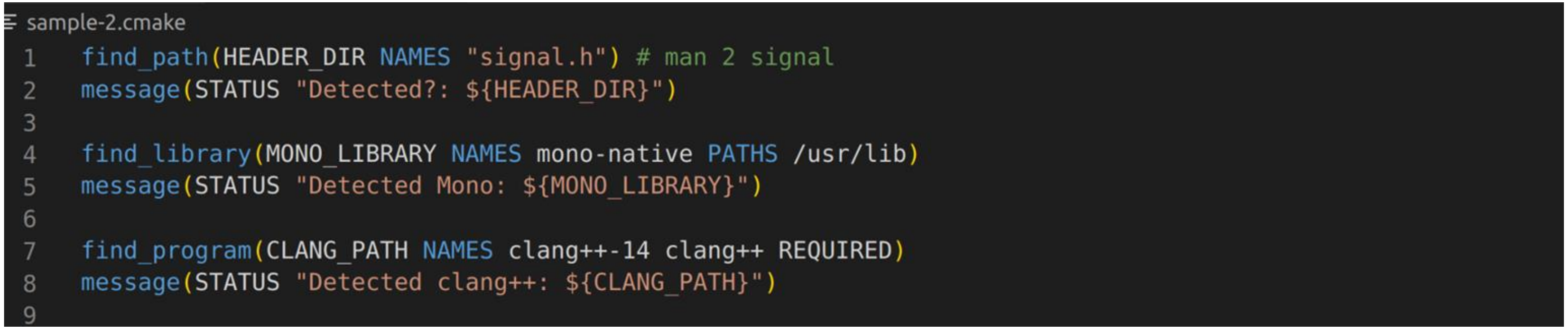
Run CMake in script mode



```
luncliff@xps-15-9550:~/Desktop/sample$ cmake -P sample.cmake
-- Say Hell World
-- Say Hell;World
```

CMake - Basic Commands

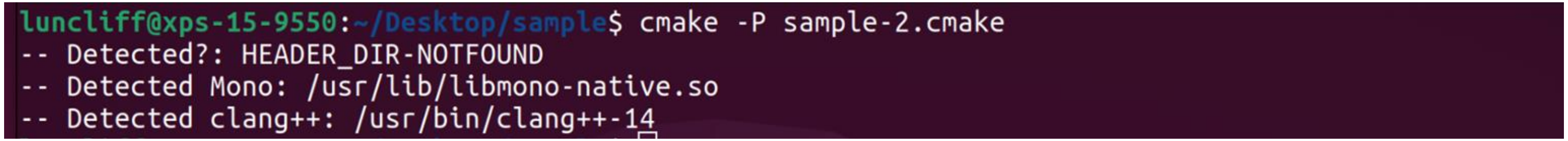
CMake provides commands find file, library, and program



sample-2.cmake

```cmake
find_path(HEADER_DIR NAMES "signal.h") # man 2 signal
message(STATUS "Detected?: ${HEADER_DIR}")

find_library(MONO_LIBRARY NAMES mono-native PATHS /usr/lib)
message(STATUS "Detected Mono: ${MONO_LIBRARY}")

find_program(CLANG_PATH NAMES clang++-14 clang++ REQUIRED)
message(STATUS "Detected clang++: ${CLANG_PATH}")

```

And allows failure



```
luncliff@xps-15-9550:~/Desktop/sample$ cmake -P sample-2.cmake
-- Detected?: HEADER_DIR-NOTFOUND
-- Detected Mono: /usr/lib/libmono-native.so
-- Detected clang++: /usr/bin/clang++-14
```

CMake - Module & Script

There is a way to distinguish CMake files.

CMake Modules

- Named like “Find*.cmake”
- Used with **find_package** command

CMake Scripts

- “*.cmake” files (in general…)
- Used with **include** command

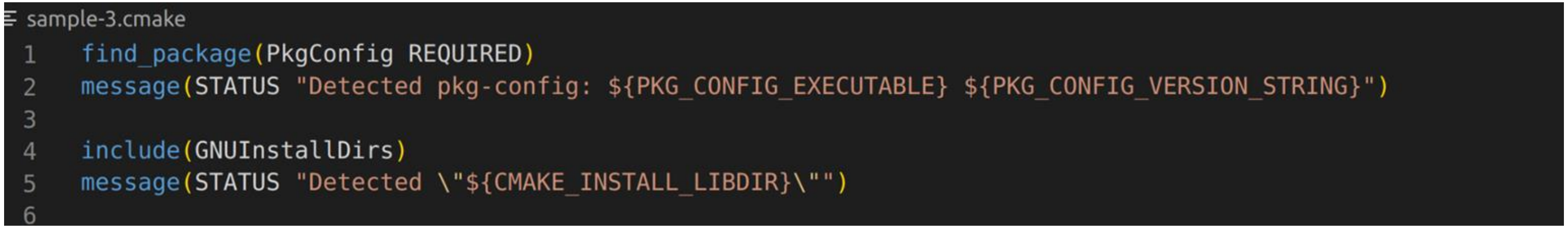
CMake - More Explanation

For example,

- https://cmake.org/cmake/help/latest/module/**FindPkgConfig**.html
- https://cmake.org/cmake/help/latest/module/**GNUInstallDirs**.html

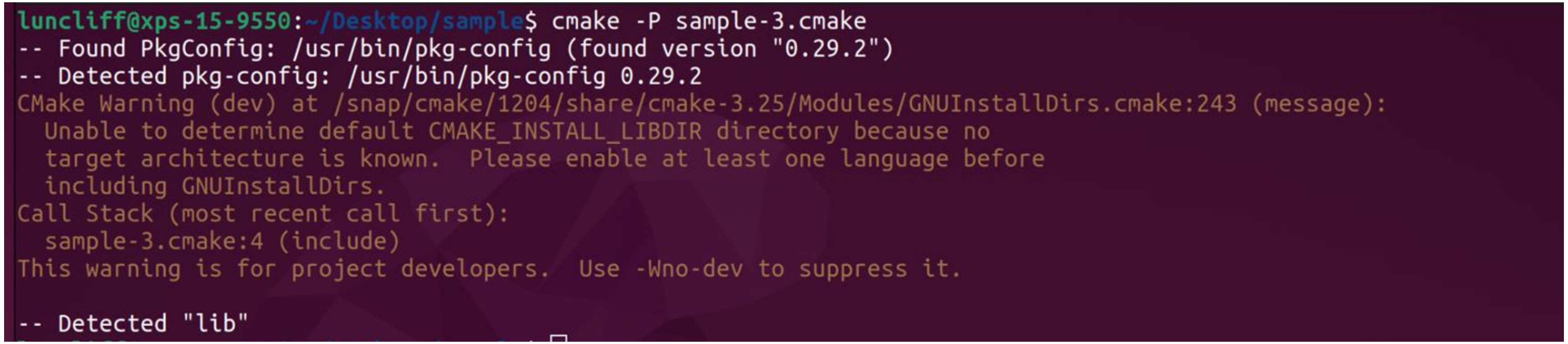
CMake - More Explanation

CMake may **warn** you if something is wrong.



sample-3.cmake

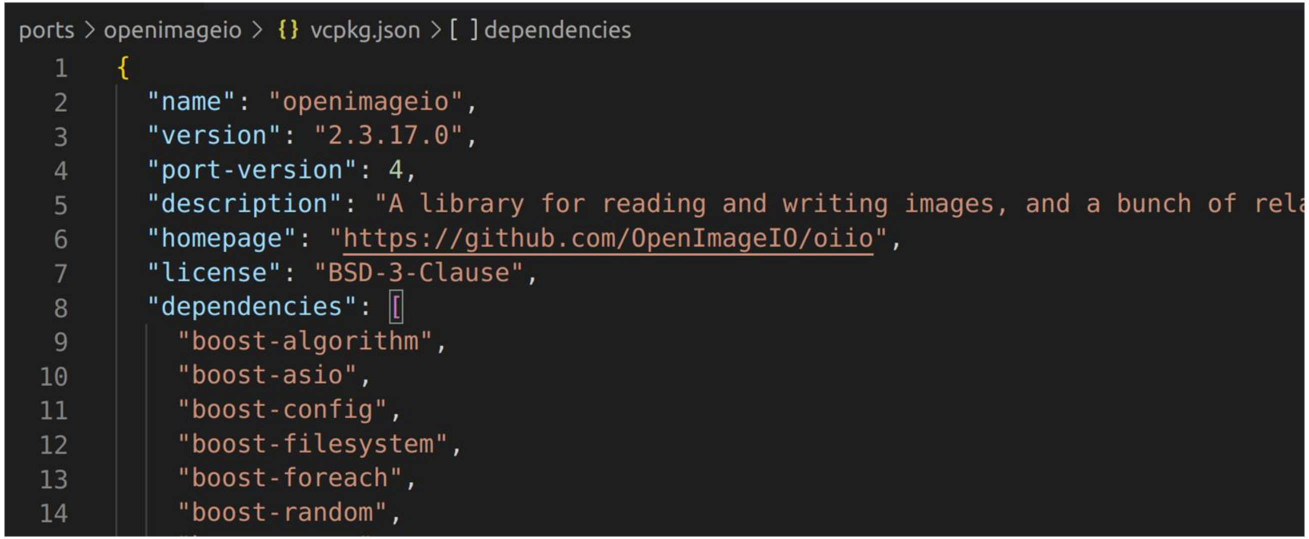
```cmake
find_package(PkgConfig REQUIRED)
message(STATUS "Detected pkg-config: ${PKG_CONFIG_EXECUTABLE} ${PKG_CONFIG_VERSION_STRING}")

include(GNUInstallDirs)
message(STATUS "Detected \"${CMAKE_INSTALL_LIBDIR}\"")

```



```
luncliff@xps-15-9550:~/Desktop/sample$ cmake -P sample-3.cmake
-- Found PkgConfig: /usr/bin/pkg-config (found version "0.29.2")
-- Detected pkg-config: /usr/bin/pkg-config 0.29.2
CMake Warning (dev) at /snap/cmake/1204/share/cmake-3.25/Modules/GNUInstallDirs.cmake:243 (message):
  Unable to determine default CMAKE_INSTALL_LIBDIR directory because no
  target architecture is known.  Please enable at least one language before
  including GNUInstallDirs.
Call Stack (most recent call first):
  sample-3.cmake:4 (include)
This warning is for project developers.  Use -Wno-dev to suppress it.

-- Detected "lib"
```

Importing Port - Case 1

Dependent program(library) generation should be listed in vcpkg.json



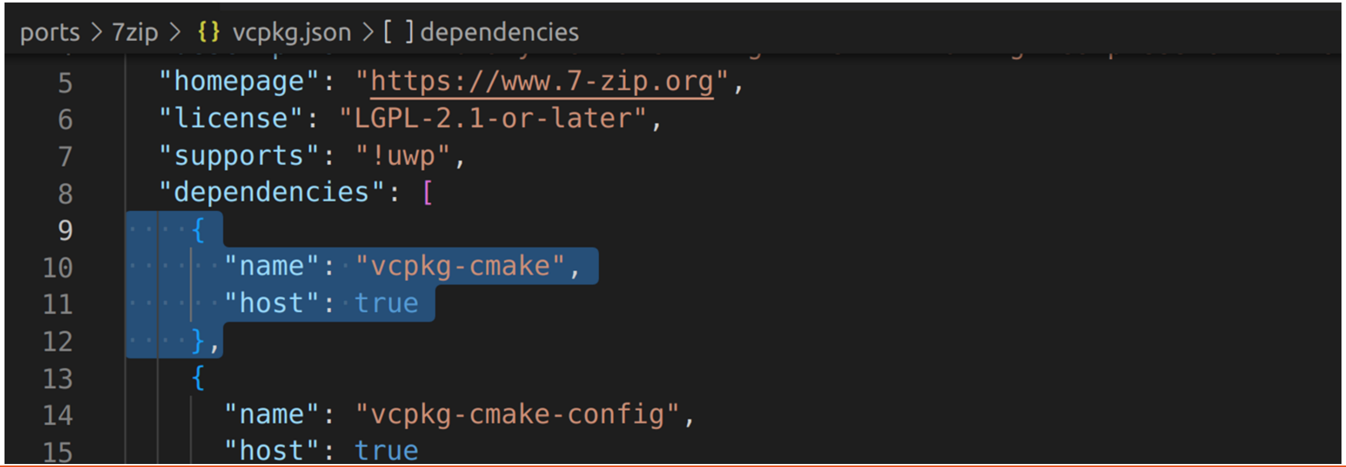
ports > openimageio > {} vcpkg.json > [ ] dependencies

```
{
  "name": "openimageio",
  "version": "2.3.17.0",
  "port-version": 4,
  "description": "A library for reading and writing images, and a bunch of rela
  "homepage": "https://github.com/OpenImageIO/oiio",
  "license": "BSD-3-Clause",
  "dependencies": [
    "boost-algorithm",
    "boost-asio",
    "boost-config",
    "boost-filesystem",
    "boost-foreach",
    "boost-random",
```

Importing Port - Case 2

Host Dependencies contain scripts, tools to run on build environment

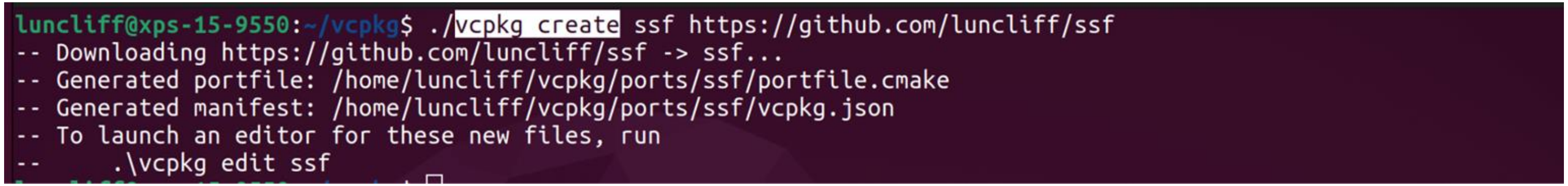
- vcpkg-cmake → for CMake project
- vcpkg-tool-meson → for Meson project
- vcpkg-get-python-packages → You have to work with Python


ports > 7zip > {} vcpkg.json > [ ] dependencies

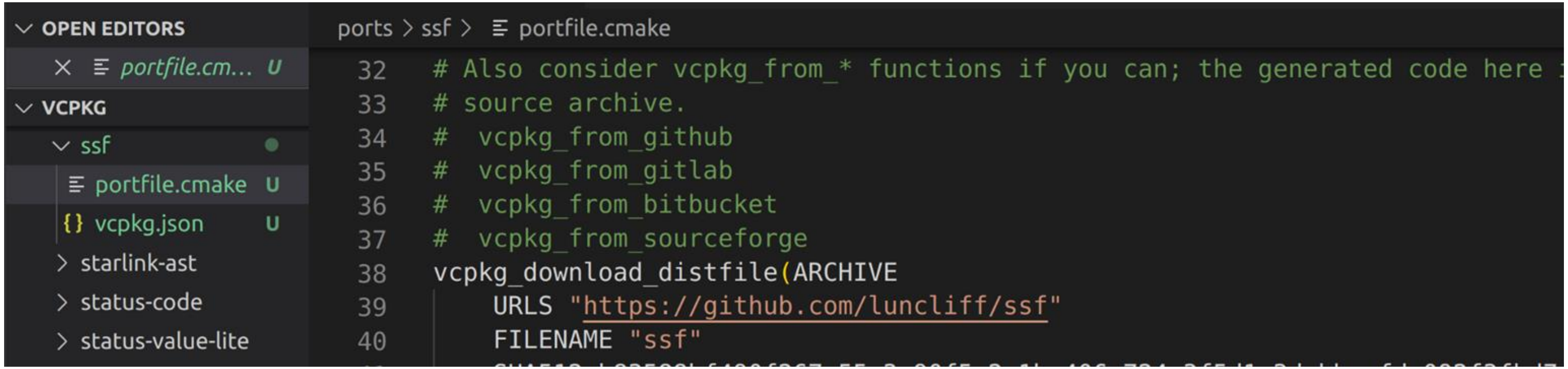
```
  "homepage": "https://www.7-zip.org",
  "license": "LGPL-2.1-or-later",
  "supports": "!uwp",
  "dependencies": [
    {
      "name": "vcpkg-cmake",
      "host": true
    },
    {
      "name": "vcpkg-cmake-config",
      "host": true
```

Port Writing - Create A New One

vcpkg already supports a command for this!



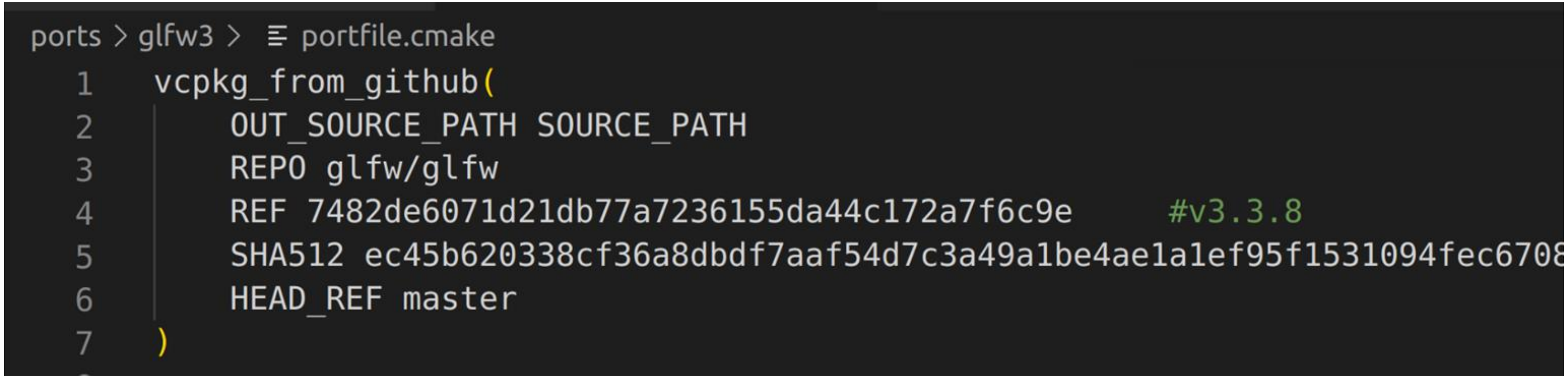
```
luncliff@xps-15-9550:~/vcpkg$ ./vcpkg create ssf https://github.com/luncliff/ssf
-- Downloading https://github.com/luncliff/ssf -> ssf...
-- Generated portfile: /home/luncliff/vcpkg/ports/ssf/portfile.cmake
-- Generated manifest: /home/luncliff/vcpkg/ports/ssf/vcpkg.json
-- To launch an editor for these new files, run
--      .\vcpkg edit ssf
```



```
# Also consider vcpkg_from_* functions if you can; the generated code here
# source archive.
#  vcpkg_from_github
#  vcpkg_from_gitlab
#  vcpkg_from_bitbucket
#  vcpkg_from_sourceforge
vcpkg_download_distfile(ARCHIVE
    URLS "https://github.com/luncliff/ssf"
    FILENAME "ssf"
```

Port Writing - Source Download 1/3

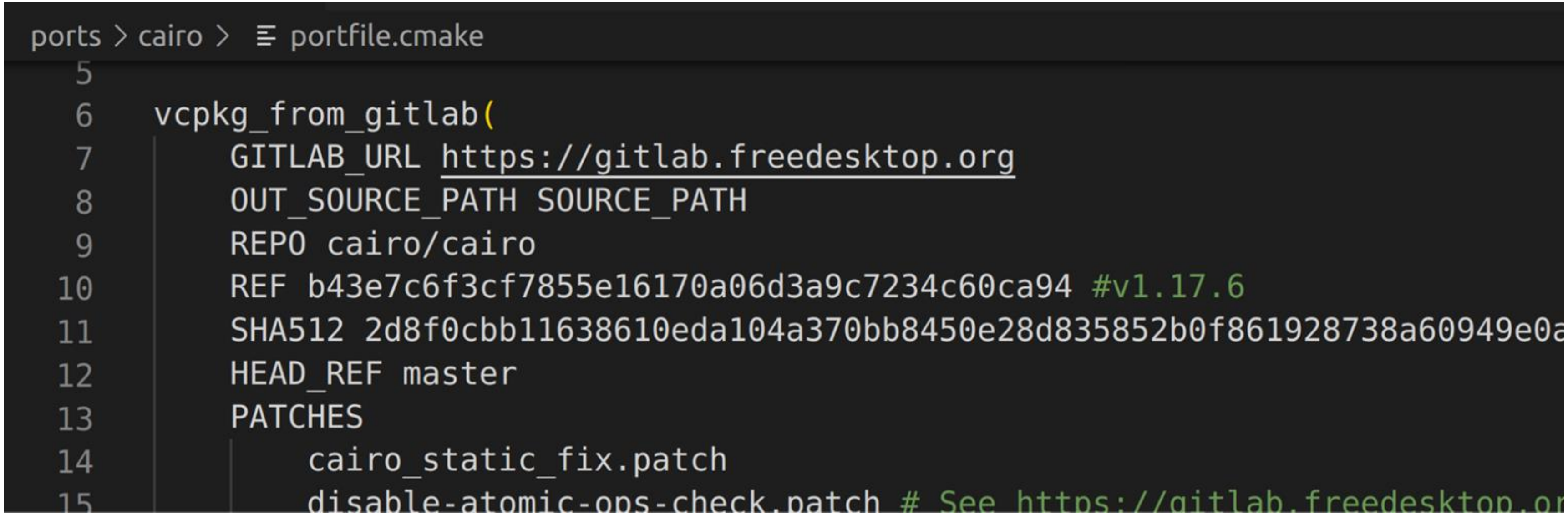
From GitHub, GitHub Enterprise


ports > glfw3 > portfile.cmake

```
vcpkg_from_github(
    OUT_SOURCE_PATH SOURCE_PATH
    REPO glfw/glfw
    REF 7482de6071d21db77a7236155da44c172a7f6c9e     #v3.3.8
    SHA512 ec45b620338cf36a8dbdf7aaf54d7c3a49a1be4ae1a1ef95f1531094fec6708
    HEAD_REF master
)
```

Port Writing - Source Download 2/3

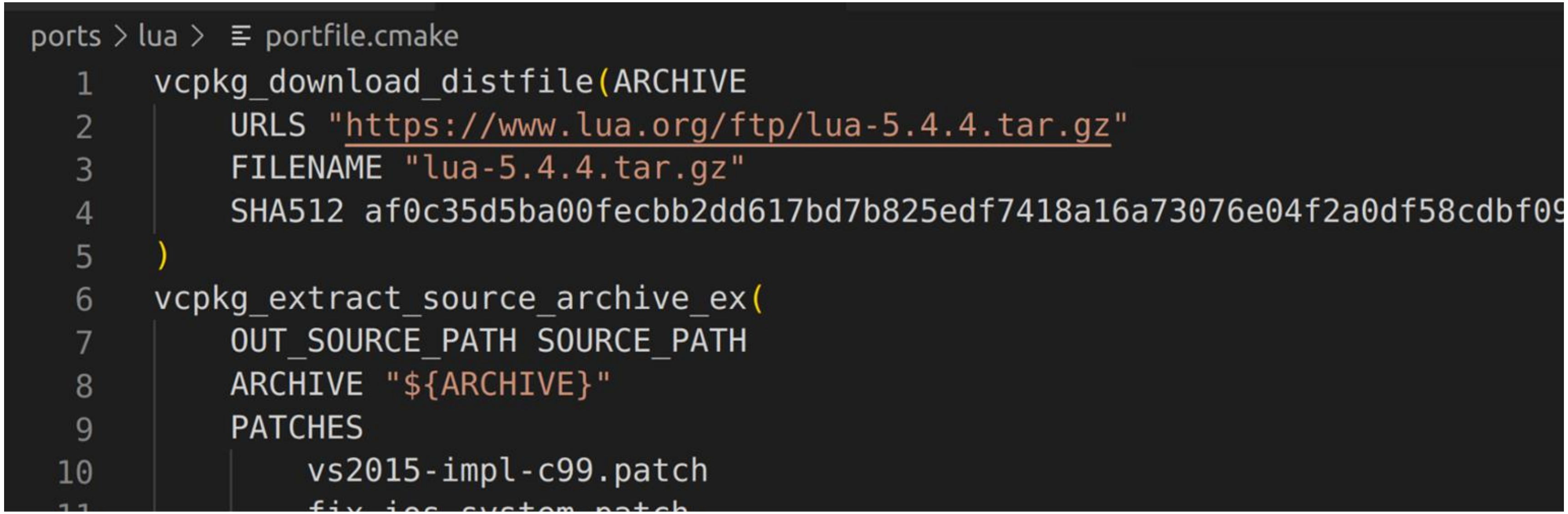
From GitLab


ports > cairo > portfile.cmake

```

vcpkg_from_gitlab(
    GITLAB_URL https://gitlab.freedesktop.org
    OUT_SOURCE_PATH SOURCE_PATH
    REPO cairo/cairo
    REF b43e7c6f3cf7855e16170a06d3a9c7234c60ca94 #v1.17.6
    SHA512 2d8f0cbb11638610eda104a370bb8450e28d835852b0f861928738a60949e0a
    HEAD_REF master
    PATCHES
        cairo_static_fix.patch
        disable-atomic-ops-check.patch # See https://gitlab.freedesktop.or
```

Port Writing - Source Download 3/3

From Non-Git

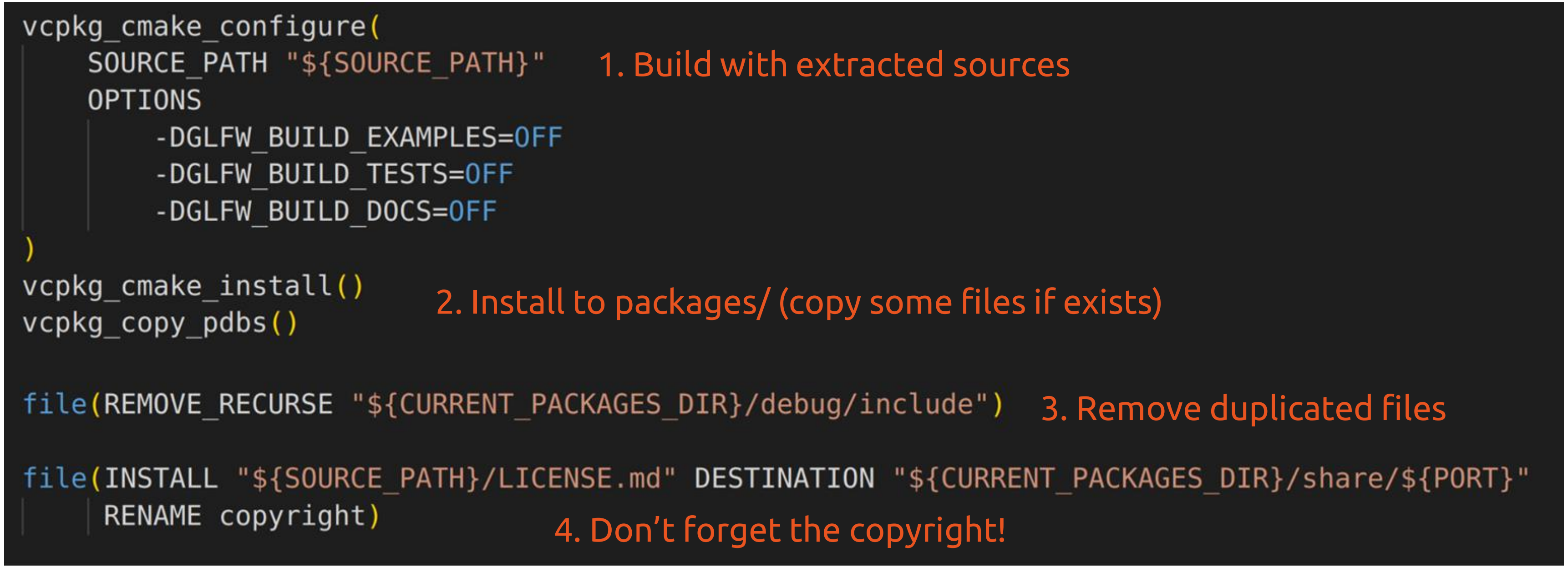


ports > lua > portfile.cmake

```
vcpkg_download_distfile(ARCHIVE
    URLS "https://www.lua.org/ftp/lua-5.4.4.tar.gz"
    FILENAME "lua-5.4.4.tar.gz"
    SHA512 af0c35d5ba00fecbb2dd617bd7b825edf7418a16a73076e04f2a0df58cdbf09
)
vcpkg_extract_source_archive_ex(
    OUT_SOURCE_PATH SOURCE_PATH
    ARCHIVE "${ARCHIVE}"
    PATCHES
        vs2015-impl-c99.patch
        fix-ios-system.patch
```

Port Writing - Configure/Install

Mostly you can just copy a similar ports and make partial changes.

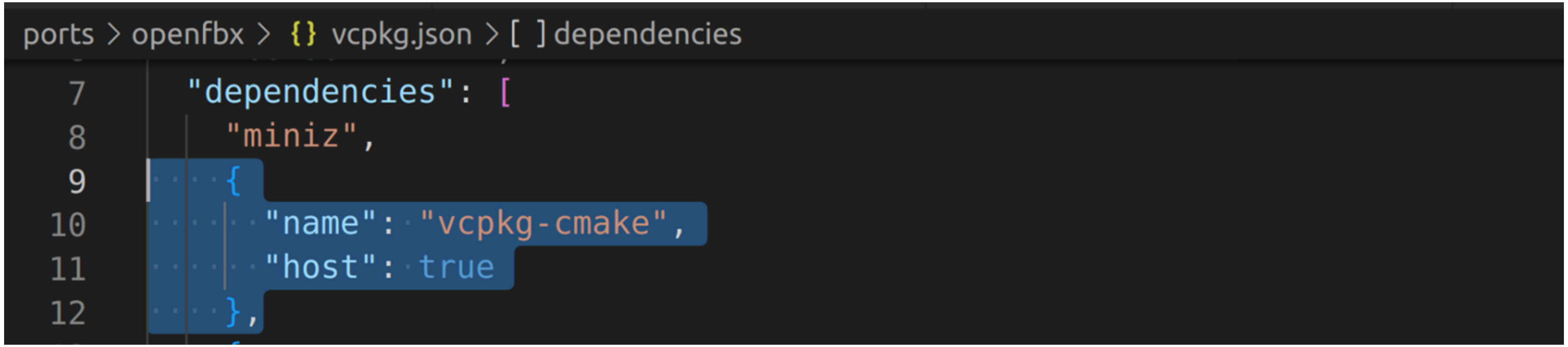


```
vcpkg_cmake_configure(
    SOURCE_PATH "${SOURCE_PATH}"
    OPTIONS
        -DGLFW_BUILD_EXAMPLES=OFF
        -DGLFW_BUILD_TESTS=OFF
        -DGLFW_BUILD_DOCS=OFF
)
vcpkg_cmake_install()
vcpkg_copy_pdbs()

file(REMOVE_RECURSE "${CURRENT_PACKAGES_DIR}/debug/include")

file(INSTALL "${SOURCE_PATH}/LICENSE.md" DESTINATION "${CURRENT_PACKAGES_DIR}/share/${PORT}"
    RENAME copyright)
```

1. Build with extracted sources
2. Install to packages/ (copy some files if exists)
3. Remove duplicated files
4. Don't forget the copyright!

Port Writing - With CMake

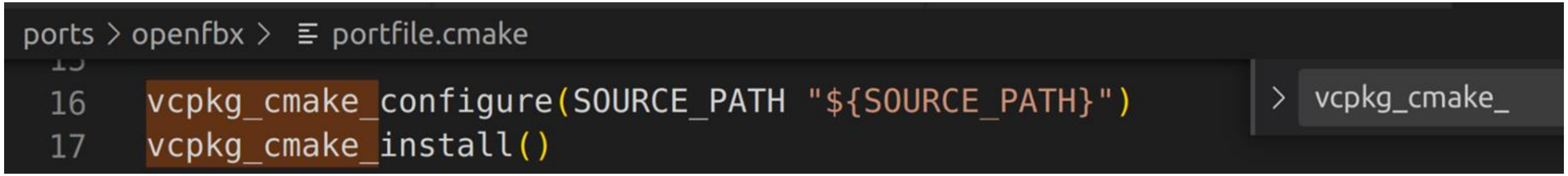
1. Host dependency “vcpkg-cmake”



ports > openfbx > {} vcpkg.json > [ ] dependencies

```
  "dependencies": [
    "miniz",
    {
      "name": "vcpkg-cmake",
      "host": true
    },
```

2. configure → cmake

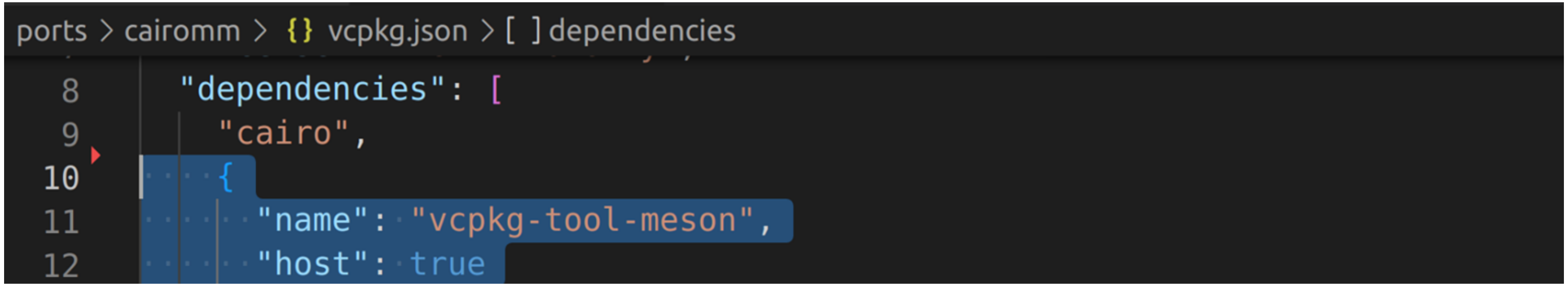


ports > openfbx > portfile.cmake

```
vcpkg_cmake_configure(SOURCE_PATH "${SOURCE_PATH}")
vcpkg_cmake_install()
```

Port Writing - With Meson

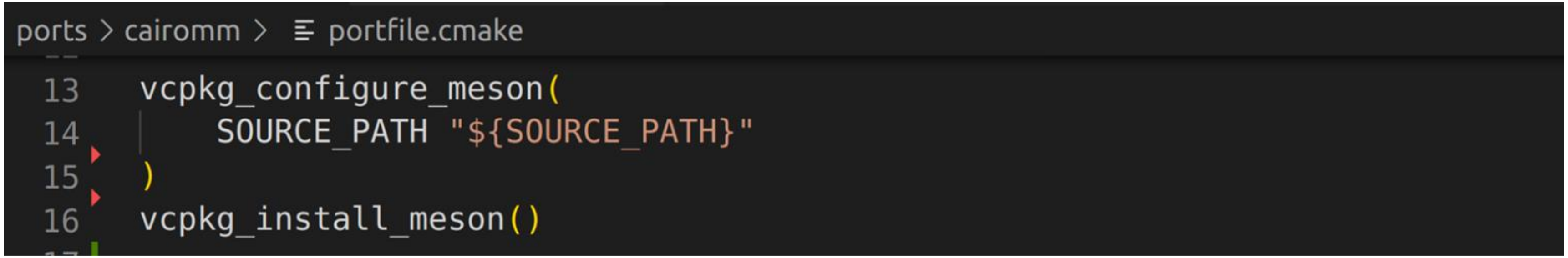
1. Host dependency “vcpkg-tool-meson”



ports > cairomm > {} vcpkg.json > [ ] dependencies

```
  "dependencies": [
    "cairo",
    {
      "name": "vcpkg-tool-meson",
      "host": true
```

2. configure → install

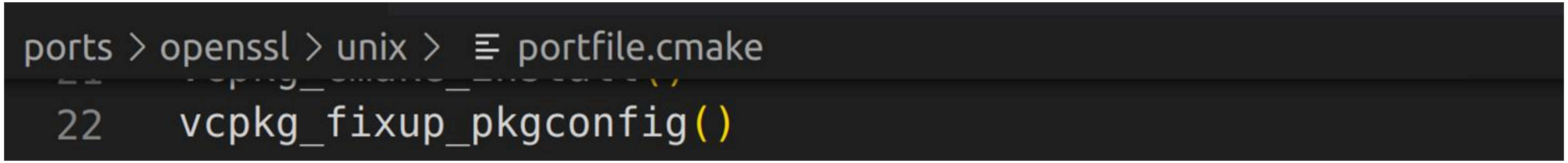


ports > cairomm > portfile.cmake

```
vcpkg_configure_meson(
    SOURCE_PATH "${SOURCE_PATH}"
)
vcpkg_install_meson()
```

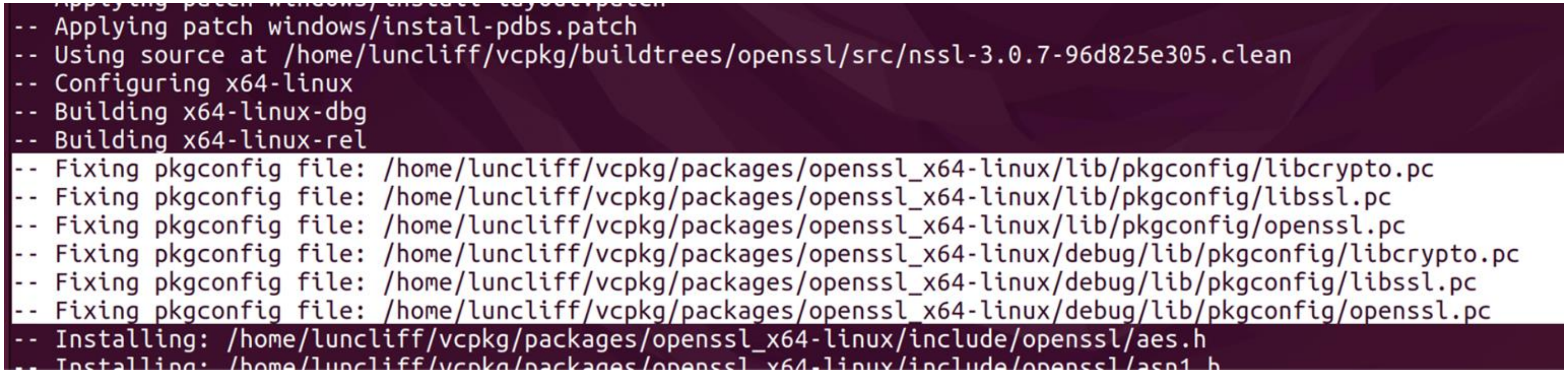
Port Writing - Fixup

Sometimes, a step so called “fixup” is required.
Projects that support **pkg-config** integration may need to do this.



ports > openssl > unix > portfile.cmake

```
vcpkg_fixup_pkgconfig()
```



```
-- Applying patch windows/install-pdbs.patch
-- Using source at /home/luncliff/vcpkg/buildtrees/openssl/src/nssl-3.0.7-96d825e305.clean
-- Configuring x64-linux
-- Building x64-linux-dbg
-- Building x64-linux-rel
-- Fixing pkgconfig file: /home/luncliff/vcpkg/packages/openssl_x64-linux/lib/pkgconfig/libcrypto.pc
-- Fixing pkgconfig file: /home/luncliff/vcpkg/packages/openssl_x64-linux/lib/pkgconfig/libssl.pc
-- Fixing pkgconfig file: /home/luncliff/vcpkg/packages/openssl_x64-linux/lib/pkgconfig/openssl.pc
-- Fixing pkgconfig file: /home/luncliff/vcpkg/packages/openssl_x64-linux/debug/lib/pkgconfig/libcrypto.pc
-- Fixing pkgconfig file: /home/luncliff/vcpkg/packages/openssl_x64-linux/debug/lib/pkgconfig/libssl.pc
-- Fixing pkgconfig file: /home/luncliff/vcpkg/packages/openssl_x64-linux/debug/lib/pkgconfig/openssl.pc
-- Installing: /home/luncliff/vcpkg/packages/openssl_x64-linux/include/openssl/aes.h
-- Installing: /home/luncliff/vcpkg/packages/openssl_x64-linux/include/openssl/asn1.h
```

Demo 2 - Reading/Writing A Port

There will be a question time after this!

1. zlib-ng with CMake
   - Check vcpkg.json - Host dependencies, Dependencies
   - Check portfile.cmake - Functions for cmake build/install
   - Fixup messages in CLI
2. libxmlpp with Meson
   - Check portfile.cmake - Functions for meson build/install
   - Compare packages/ folder
3. openssl
   - Running pkg-config in CLI
   - Buildtree comparison

Part 3: Diagnosing Log Files

Logs from package installation

Possible errors in Ubuntu

Checklist - Before Log Analysis

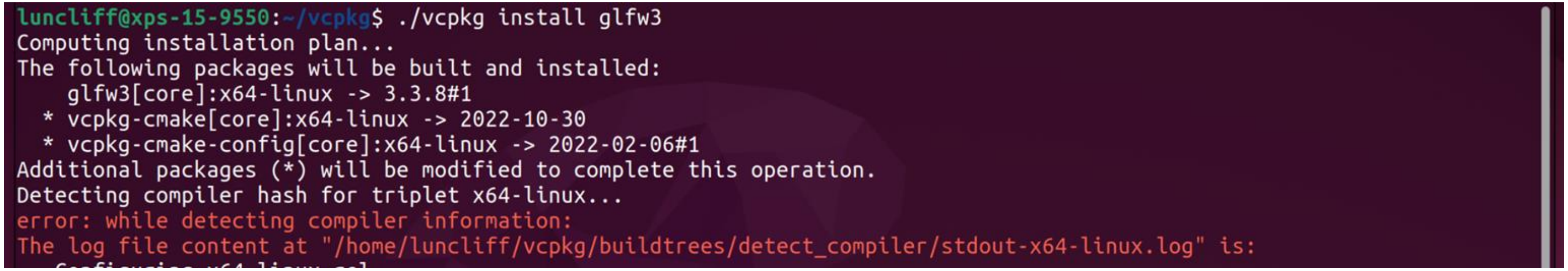
Be familiar with Top-Down approach. Check high level to low level.

1. Environment trouble
   - PATH environment variable
   - Some variables may affect the tool behavior.
     ex) https://cmake.org/cmake/help/latest/manual/cmake-env-variables.7.html
2. Vcpkg's commit ID and the project CI status
3. Triplet files
   - VCPKG_ variables can be bad(wrong)
4. Port, Patch files
   - portfile.cmake is not using variables from the triplet
   - Wrong source version, Mistake in the patch file

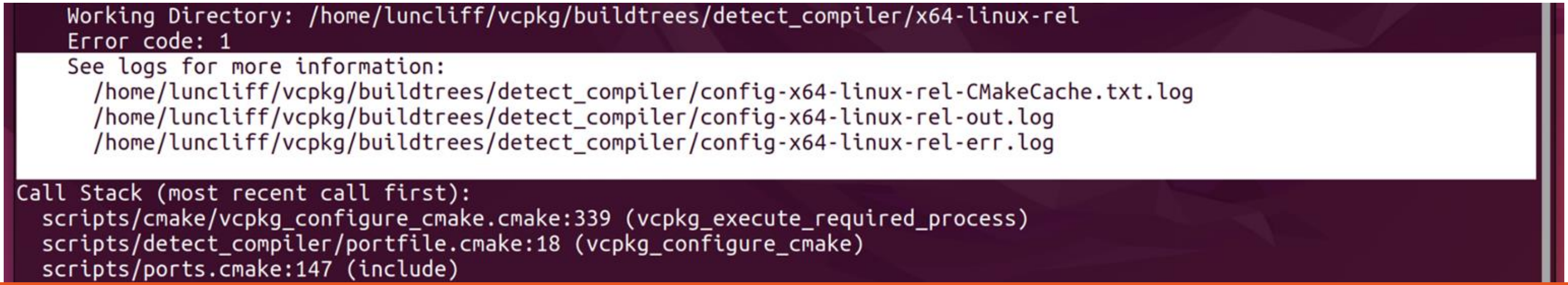
Checklist - When Failure Encountered

Q. If install failed, what should we check first?

A. Vcpkg will report you!



```
luncliff@xps-15-9550:~/vcpkg$ ./vcpkg install glfw3
Computing installation plan...
The following packages will be built and installed:
    glfw3[core]:x64-linux -> 3.3.8#1
  * vcpkg-cmake[core]:x64-linux -> 2022-10-30
  * vcpkg-cmake-config[core]:x64-linux -> 2022-02-06#1
Additional packages (*) will be modified to complete this operation.
Detecting compiler hash for triplet x64-linux...
error: while detecting compiler information:
The log file content at "/home/luncliff/vcpkg/buildtrees/detect_compiler/stdout-x64-linux.log" is:
```



```
    Working Directory: /home/luncliff/vcpkg/buildtrees/detect_compiler/x64-linux-rel
    Error code: 1
    See logs for more information:
      /home/luncliff/vcpkg/buildtrees/detect_compiler/config-x64-linux-rel-CMakeCache.txt.log
      /home/luncliff/vcpkg/buildtrees/detect_compiler/config-x64-linux-rel-out.log
      /home/luncliff/vcpkg/buildtrees/detect_compiler/config-x64-linux-rel-err.log

Call Stack (most recent call first):
  scripts/cmake/vcpkg_configure_cmake.cmake:339 (vcpkg_execute_required_process)
  scripts/detect_compiler/portfile.cmake:18 (vcpkg_configure_cmake)
  scripts/ports.cmake:147 (include)
```

Checklist - Check Host Environment Changes

Remind the vcpkg characteristics!

1. Build from sources
2. Works in 1 folder
3. Environment variable can affect behavior

Guess what…

1. Build tools are updated (by APT or Snap)
2. The vcpkg root contains garbage
3. Vcpkg executables and ports are updated
4. VCPKG_ environment variables are defined in .bashrc or shell session

Checklist - Well-Known Log Names

- stdout-${triplet}.log
  - Entire messages in CLI
- error-logs-${triplet}.txt
  - Files that may contain details of the error
- config-${triplet}-dbg/rel-out.log
  - Buildsystem file generation failure details
- build/install-${triplet}.log
  - Executable Troubles. Mostly toolchain.
  - Build messages - Compiler mismatch & Linker errors

Log Analysis - Buildsystem File Generation

Failure Patterns

1. Host dependencies need update
2. Package, Library search
3. CMake/Meson is making unexpected behaviors/errors
    - You may have to create a .patch for the port…
4. System package manager is interrupting Vcpkg

Log Analysis - Port Build/Install

Buildsystem logs are not that much readable.

Compiler/Linker arguments matters because it's C/C++ world.

- Macro definitions
- Options
- Warning messages

Sometimes compiler may die with ICE (Internal Compiler Error).
In this case you have to change the build toolchain.

Reporting - Use GitHub Repository

The channel can be time consuming

1. Search the existing Issues/Discussions
2. Search the similar error messages
3. **Ask questions who created the port**
   (Or ask the person who knows or can build it!)
4. Check the instructions of the library
   (portfile.cmake can have a bug)

Demo 3 - Reading Success/Failure Logs

There will be a question time after this!

1. Successful install
2. Triplet failure
3. Port failure
4. Files to check before reporting
5. Reviewing existing reports in GitHub

Summary

What can we do after this workshop?

What Should We Remember?

Vcpkg characteristics

Variables that **Triplets** should define

- VCPKG_ variables (Check the docs/)

Port structure

- (Host) dependencies in vcpkg.json
- Source download → extract → configure → install → fixup

How to check **Log** files

What Can I Learn After This?

Read the manuals under docs/users/.

Vcpkg features for…

- Deploying/Sharing files with others
- Accessing private resources (Authentication issues)

Contribution

- Share your work in upstream (Pull Requests)
- Answer/Vote to existing discussions

Thank you!

For more questions, you can send a mail to me!
luncliff@gmail.com / luncliff@cppkorea.org

Or create an issue at github.com/luncliff/vcpkg-registry