LAAS CNRS

CMake Wheel

Gérez vos projects CMake et leurs dépendances avec pip

2022-11-20

Guilhem Saurel

cnrs

Université Fédérale Toulouse Midi-Pyrénées

LAAS CNRS

This presentation

Available at

`https://nim65s.github.io/talks/cmake-wheel.pdf`

Under License

https://creativecommons.org/licenses/by-sa/4.0/

This presentation (continued)

Source

`https://github.com/nim65s/talks : cmake-wheel.md`

Discussions

https://matrix.to/#/room/#cmake-wheel:matrix.org

CMake

- C / C++ (et plus si affinitées)
- GCC / LLVM / MSVC / Intel / ...
- Linux / *BSD / macOS / Windows / ...

Figure 1: CMake

Modern CMake

```
find_package(Boost REQUIRED COMPONENTS regex)
add_library(pipo SHARED pipo.hpp pipo.cpp)
include_directories(${Boost_INCLUDE_DIRS})
target_link_libraries(pipo PUBLIC
    ${Boost_LIBRARIES})
```

Modern CMake

```
find_package(Boost REQUIRED COMPONENTS regex)
add_library(pipo SHARED pipo.hpp pipo.cpp)
include_directories(${Boost_INCLUDE_DIRS})
target_link_libraries(pipo PUBLIC
    ${Boost_LIBRARIES})

-include_directories(${Boost_INCLUDE_DIRS})
-target_link_libraries(pipo PUBLIC
-    ${Boost_LIBRARIES})
+target_link_libraries(pipo PUBLIC Boost::regex)
```

Modern CMake

```
find_package(Boost REQUIRED COMPONENTS regex)
add_library(pipo SHARED pipo.hpp pipo.cpp)
include_directories(${Boost_INCLUDE_DIRS})
target_link_libraries(pipo PUBLIC
    ${Boost_LIBRARIES})

-include_directories(${Boost_INCLUDE_DIRS})
-target_link_libraries(pipo PUBLIC
-    ${Boost_LIBRARIES})
+target_link_libraries(pipo PUBLIC Boost::regex)
```

=> Paquets relocalisables

pip

```
python -m pip install --user django
```

PEP 517

A build-system independent format for source trees

Permet à n'importe qui de créer un système de build en écrivant une fonction:

```
def build_wheel(
        wheel_directory,
        config_settings=None,
        metadata_directory=None,
):
    ...
```

CMake Wheel

```python
from subprocess import run

def build_wheel(...):
    run("cmake -S src -B build")
    run("cmake --build build")
    run("cmake --build build -t test")
    run("cmake --install build")
```

Wheel (PEP 427)

A wheel is a ZIP-format archive with a specially formatted file name and the .whl extension.

Wheel (PEP 427)

> *A wheel is a ZIP-format archive with a specially formatted file name and the .whl extension.*

eg: `hpp_fcl-2.1.3-4-cp310-cp310-manylinux_2_28_x86_64.whl`

Config: pyproject.toml (PEP 518)

```toml
[project]
name = "cmeel-example"
version = "0.2.3"
description = "Example project"
readme = "README.md"
license = "BSD-2-Clause"
authors = ["guilhem.saurel@laas.fr"]

[project.urls]
repository = "https://github.com/cmake-wheel/cmeel-ex

[build-system]
requires = ["cmeel[build]"]
build-backend = "cmeel.build"
```

Use

- `python -m pip install \ git+https://github.com/cmake-wheel/cmeel-example.git`
- `python -m pip install cmeel-example`

Use

- `python -m pip install \ git+https://github.com/cmake-wheel/cmeel-example.git`
- `python -m pip install cmeel-example`
- `cmeel-add 3 4`
- `python -c "import cmeel_example; print(cmeel_example.cmeel_add(3, 4))"`

Publish your packages to PyPI from CI

with cibuildwheel and github actions for:

- CPython 3.7 - 3.11 / pypy 3.7 - 3.9
- manylinux / musllinux / macos
- x86_64 / aarch64 / ppc64le / i686 / s390x

And cirrus-CI for Apple Silicon

Design

- `-DCMAKE_INSTALL_PREFIX=${SITELIB}/cmeel.prefix`
- `${SITELIB}/cmeel.pth`
- `…/cmeel.prefix/bin => cmeel.run:cmeel_run`
- helpers for `$CMAKE_PREFIX_PATH`, `$LD_LIBRARY_PATH`, …

Design

- `-DCMAKE_INSTALL_PREFIX=${SITELIB}/cmeel.prefix`
- `${SITELIB}/cmeel.pth`
- `…/cmeel.prefix/bin => cmeel.run:cmeel_run`
- helpers for `$CMAKE_PREFIX_PATH`, `$LD_LIBRARY_PATH`, …

Contraintes:

- CMake
- paquets relocalisables
- `RPATH` / `@loader_path`

References

- https://github.com/cmake-wheel/cmeel
- https://github.com/cmake-wheel/cmeel-example
- https://cmeel.rtfd.io

C’est tout pour moi !

La parole est à vous :)