Introduction to the Meson build system

Code coffee talk

León-Alexander Hühn

Institute for Theoretical Astrophysics
Center for Astronomy of
Heidelberg University

22nd April, 2024

Contents

Introduction

What is a build system?

- Most basic definition: A script containing commands to compile source files and link objects into an executable
- Used for projects that have multiple source files that would be cumbersome to compile and link by hand
- Basic, low-level build system: `make`

Example: `make`

- Builds *targets* from *dependencies* using *recipes*
- Build process is defined in the `Makefile`

`Makefile`

```
main: magic.o main.c
	gcc -c main.c
	gcc -o main main.o magic.o

magic.o: magic.c
	gcc -c magic.c
```

High-level build systems

While `make` has some support for basic abstraction, it lacks many features that are useful for large and modern projects. Everything needs to be done manually:

- Determining the compiler based on availability
- Determining compiler flags based on platform and environment
- Finding library locations and determining the correct linker flags
- OS and platform-dependent packaging or installing
- ...

High-level build systems

- High-level build systems introduce an additional layer of abstraction designed to automate these tasks
- Many high-level build systems generate build files for low-level build systems instead of performing the build themselves
- Popular example for C/C++ projects: `CMake`
- New, more modern example: `Meson`

Why Meson?

. . . and not `CMake`?

- While `CMake` is very powerful, it is also very complex and has a steep learning curve
- If you are just trying to build your scientific code, you probably don't need all the features `CMake` provides, and learning it is a large time commitment

Why Meson?

Reasons to use `Meson`:

- Aims to be as user-friendly as possible, while still providing most features and being fast
- Written in Python, featuring modern-style, easy to read and learn syntax
- Interfaces well with `pip` to create Python packages

Installation

`Meson` generates build files for the low-level build system `ninja`. Both need to be installed in addition to any compilers and libraries you want to use.

- Debian derivatives: `# apt install meson ninja-build`
- MacOS: `$ brew install meson ninja`
- PyPI: `$ pip install --user meson`

Note: If you only want to use `Meson` to build Python packages, no installation is necessary. `pip` will automatically install `Meson` as a dependency during the build process.

Examples

Building a simple C program

To get to know the basic concepts of `Meson`, follow the Tutorial on the `Meson` website to create a simple C program using GTK.

`Meson` uses `meson.build` in the root directory to determine how to build the project.

The build takes requires a separate build directory: `meson setup <builddir>`.

To start the build:

- `meson compile` in the build directory or
- `meson compile -C <builddir>` in the root directory

Further useful functions

Despite its simplicity, `Meson` is quite feature rich. Some useful functions to look up in the documentation include:

- `add_project_arguments`: Add custom compiler flags
- `build_target`: Build different kinds of binaries
  - Dynamic variant of `executable`, `shared_library`, `static_library`, ...
- `custom_target`: Run custom commands
- `find_program`: Find non-compiler executables on the system
- `subdir`: Enter subdirectory and execute its `meson.build` file

Using Meson for your mixed-language Python project

- `Meson` can be used to build Python packages using its Python module (docs)
- Especially useful for projects that use a mix of pure Python and compiled modules written in C or Fortran
- Can be used as a replacement for `numpy.distutils` (removed in Python 3.12+)
  - Large projects like `scipy` have migrated to it
- Note: Migrating from `numpy.distutils` to `setuptools` is likely easier if your project only uses pure Python code

Conclusion

- `Meson` is a modern, easy to use build system
- It is a good alternative to `CMake` for scientific projects if you don't have prior experience with other build systems
- It can offer a good alternative to the deprecated `numpy.distutils` for mixed-language Python projects