CMake basics and practices For Fortran applications

Yes, it is awful but would you rather use make?

Jorge

CMake basics and practices For Fortran applications

Table of Contents

Building and deploying an application

An application is useful when: - It solves a known problem - It is easy to use - It is well documented

An application can only be truly useful if it is easy to deploy and: - It is portable across architectures - It can leverage unknown environments to the developer (some rando deploying on an IBM mainframe) - It can be customised - It can be modular

---

The Inexorable Complexity of Scientific Software

Scientific software is highly complex: - It is mostly developed by academics (students, grad students, postdocs) - Most of the developers have little skills in software engineering - The academic timeline is awful for proper practices

This results in: - Software that solves a problem specific to a group - Many repeated efforts and thus wasted time/money - Ease of use most of the times goes to the back burner - Struggling is seen as a rite of passage

---

The awfulness of build systems

- All complex applications need to be built for them to be usable
- How you compile all the files that pertain the project with a specific compiler/runtime environment is known as the "build system"

Options: - **GNU Make** — old, reliable, complex for large projects - **CMake** — newish, polarising, adopted by large companies, provides cross-compilation support - **Meson** — very new, used it once, did not like it, found a bug - **Package managers** — built into the language/framework, designed to build projects within them - Cargo (Rust) - NPM (JavaScript)

Why not just Make?

A lot of LARGE projects use Make, why not just keep it?

Reasons: - Make is not portable across architectures (Windows, Mac, Linux, ARM, IBM, etc.) — it has flaws - Supporting a wide variety of compilers is hard - Can be hacked quite freely and lead to very wacky configurations

Fortran specific reasons: - Modules and module dependencies - Ease of parallel build setups

---

CMake as a stopgap solution while we get a package manager

- The software industry has adopted CMake as a standard and as their main build system
- Game devs, HFT, toolings, and some of the main hardware vendors have adopted it:
  - AMD, NVIDIA, Intel
- (Debatable) It is easier to extend
- Simpler to support multiple compilers (if done properly)
- A lot of documentation and experience in the online world

  We can drop CMake once Fortran has a package manager (in its way!)