Cetmodules and Spack-at-FNAL in 2023

Chris Green, FNAL

2023-06-06

Overview

- The current state of Cetmodules
    - Philosophy
    - Differences with cetbuildtools
    - Support for modern CMake paradigms
    - Building with Spack
- UPS -> Spack at FNAL
    - Why?
    - Migration philosophy
    - Current status
    - Remaining hurdles

Cetmodules: philosophy

- No inherent dependence on UPS.
- Basic backward compatibility with cetbuildtools.
    - cetbuildtools 8+ is a (very) thin wrapper around Cetmodules.
- Basic migration of user code should be easy; changes to support building with Spack should be adiabatic without affecting UPS-based builds.
- Support modern (CMake >=3.0) paradigms.
- Facilitate handling of transitive dependencies.
- Dependent packages need not use Cetmodules.

Cetmodules: *cf* cetbuildtools

- Deprecation of reliance on `GLOB` (hysteresis avoidance):
  - `art_make()` *vs* `art_dictionary()`, `art_make_library()`, `build_plugin()`.
- Targets *vs* CMake/environment variables.
- Improved plugin handling:
  - Generated CMake plugin functions with configure suffix, dependencies, etc. (e.g. `build_frobnicator_tool()`) available to dependents.
  - Separates plugin implementation and registration units into separate libraries to avoid ODR violation.
  - Prevents linking to registration libraries and non-linkable plugin implementations (e.g. art modules).

Cetmodules: *cf* cetbuildtools

- More flexible/automatic generation of CMake config files for use by dependents:
  - Much better handling of transitive dependencies: build-only *vs* link *vs* header-propagated dependencies.
  - New `find_package()` keyword: `EXPORT`.
- Exportable “Project Variables.”
- In the absence of environment variables from UPS, relocatability handled with
- Generation/use of `CMakePresets.cmake` to duplicate configuration from `product_deps` when not using UPS, `setup_for_development`.

Cetmodules: modern CMake paradigms

- Targets, targets, `scoped::target`s!
- No-library (`INTERFACE`) targets for fine-grained management of header-induced dependencies.
- No-link (`MODULE`) plugin libraries.
- Object-code sharing between `SHARED` and `STATIC` libraries built from the same source (`OBJECT` “libraries”).

More details: “new” CMake concepts in Cetmodules.

Cetmodules: implications for Spack

- Semi-automatic adiabatic migration path away from UPS-isms.
- Intelligent dependency reduction—minimal header-induced dependencies (e.g. via IWYU), automated transitive dependencies—simplifies Spack recipes.
- Dependency version choices delegated to Spack recipes/concretizer.

UPS -> Spack at FNAL: why?

- UPS is older[1] than my A-Levels[2]: one (1) person remaining at the lab who understands/remembers the UPS source code well enough to maintain it.
- `LD_LIBRARY_PATH` (and variants) no longer viable as a universal system for maintaining binary package relocatability (e.g. MacOS/SIP).
- High overhead for package version updates:
  - Dependency versions pegged in table file -> error prone.
  - Manual table file updates, combinatorics.
- Build instructions are not defined by UPS (though see ssibuildshims).

[1] *UPS UNIX™ Product Support FERMILAB-CONF-91/174*
[2] *a.k.a.* High School Diploma.

UPS -> Spack at FNAL: migration philosophy

- Compatibility:
  - Generate table files to allow use as a UPS product.
  - Allow some use of pre-built UPS products as externals in Spack builds.
- Maximal use of relocatable pre-built binaries in Spack build caches.
- Reproducibility: produce configuration files describing software distribution bundles *a la* `buildcfg` files for `buildFW`.
- Flexibility: allow for locally-built distributions with different versions (e.g. Geant4).
- Ease of use: turnkey installation of vetted Spack versions and scripted installation of distributions.
- Allow for multi-package software development in the context of Spack-based builds.

UPS -> Spack at FNAL: current status

- Several experiments using *ad hoc* Spack-built software on HPC (e.g. ICARUS, DUNE).
- Scripted build (interactively or via Jenkins) of a sequence of automatically-generated distribution configurations.
- Mu2e distribution based on art-suite 3.13.01 has been built successfully with GCC 12.2.0, C++17 (`e26`), including Geant4 10.7.4 with Qt-based visualization (upcoming workshop).
- AlmaLinux 9 support in progress.

UPS -> Spack at FNAL: remaining hurdles

- Minimize unwanted rebuilds due to minor recipe changes (Spack limitation).
- Minimize reliance on system packages for grid-based production (`X11`).
- Straightforward support of multiple platforms, compilers.
- CI.
- Development.