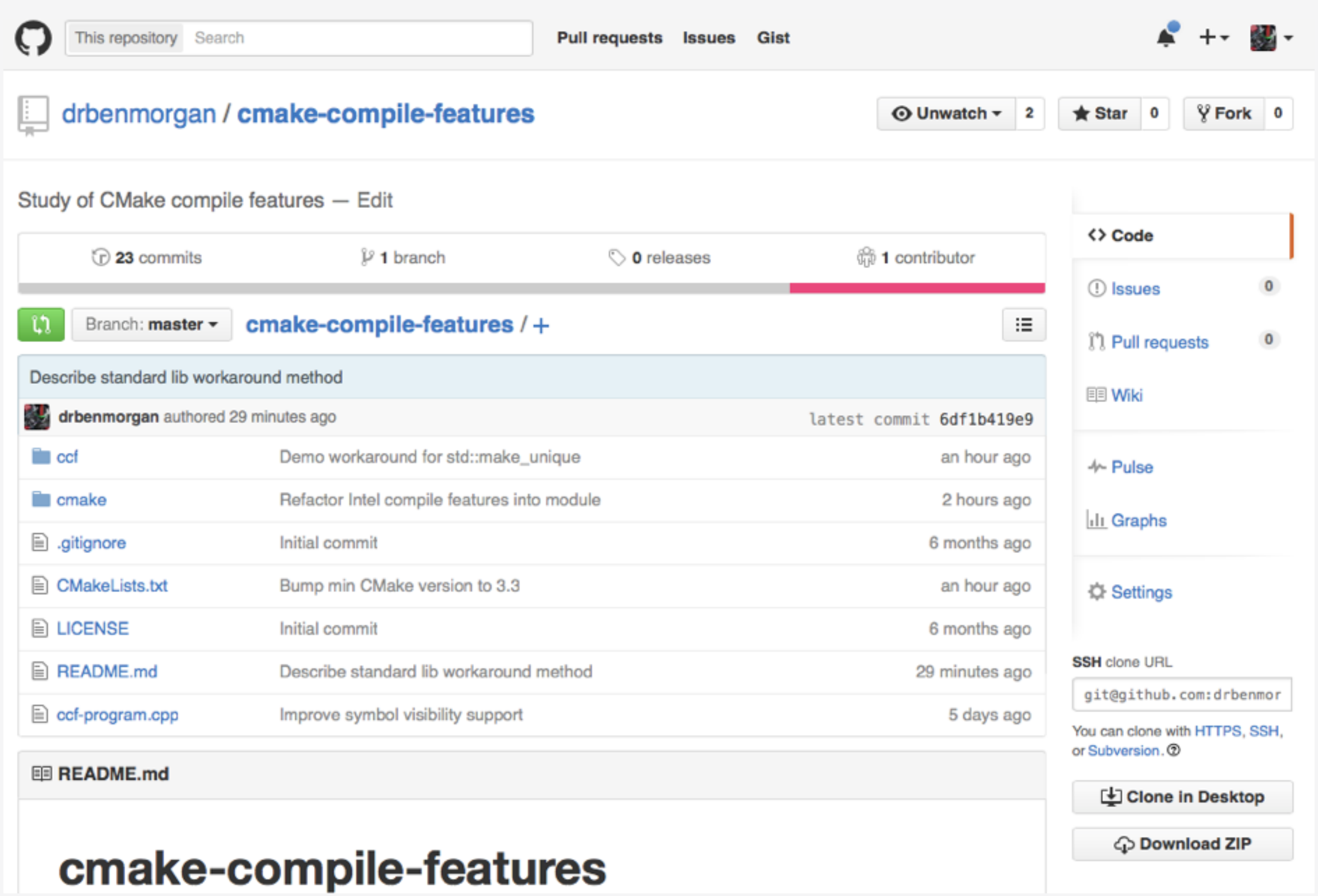
C++11/14/1z in CMake

Ben Morgan

THE UNIVERSITY OF WARWICK

Overview

- How do we ensure the compiler and standard library in use support the features of C++ Standard used in code?
- Can we provide workarounds when/if they don’t?
- How do we help users of Geant4 to compile their applications against the same standard?
- ***A (not so) short demo of using CMake 3 to solve these issues.***



drbenmorgan / cmake-compile-features

Study of CMake compile features — Edit

23 commits | 1 branch | 0 releases | 1 contributor

Branch: master | cmake-compile-features / +

Describe standard lib workaround method

drbenmorgan authored 29 minutes ago | latest commit 6df1b419e9

ccf	Demo workaround for std::make_unique	an hour ago
cmake	Refactor Intel compile features into module	2 hours ago
.gitignore	Initial commit	6 months ago
CMakeLists.txt	Bump min CMake version to 3.3	an hour ago
LICENSE	Initial commit	6 months ago
README.md	Describe standard lib workaround method	29 minutes ago
ccf-program.cpp	Improve symbol visibility support	5 days ago

SSH clone URL

git@github.com:drbenmor

You can clone with HTTPS, SSH, or Subversion.

README.md

cmake-compile-features

https://github.com/drbenmorgan/cmake-compile-features

lambdas

`[]{foo();}`

`constexpr`

initializer lists

`regex`

C++11

`nullptr`

```
shared_ptr<T>,
unique_ptr<T>,
  weak_ptr<T>
```

```
auto i = v.begin();

for(auto x : collection)
```

Syntax vs Standard Library Features

```
# CMakeLists.txt
…
include(cmake/CheckCXXFeature.cmake)
check_cxx11_feature(
  “cxx_memory_make_unique”
  HAS_CXX_MEMORY_MAKEUNIQUE
  )


…


add_library(foo foo.hpp foo.cpp)
target_compile_features(foo
  PUBLIC
    cxx_constexpr
  )
```

Compiler Exercise

Compiler Knowledge

Checking C++ Library/Language Features

```
# CMakeLists.txt
include(WriteCompilerDetectionHeader)
write_compiler_detection_header(
  FILE      foo_compiler_support.hpp
  PREFIX    FOO
  COMPILERS GNU Clang MSVC
  FEATURES  cxx_thread_local
  )

configure_file(foo_stdlib_support.hpp.in
  foo_stdlib_support.hpp
  )
```

```
// - foo_stdlib_support.hpp.in
#include <memory>
#cmakedefine HAS_CXX_MEMORY_MAKE_UNIQUE
#ifndef HAS_CXX_MEMORY_MAKE_UNIQUE
… local implementation of std::make_unique …
#endif
```

Working Around Missing Features: CMake

```
// - foo.cpp
#include “foo_compiler_support.hpp”
#include “foo_stdlib_support.hpp”
…
CCF_THREAD_LOCAL myTLVariable;

void someFunction() {
  auto fooPtr = std::make_unique<Foo>();
  …
}
```

Working Around Missing Features: Code

```
# CMakeLists.txt for foo project
add_library(foo foo.hpp foo.cpp)
target_compile_features(foo
  PUBLIC
    cxx_constexpr
  )
```

"c++ -std=c++11 … libfoo.so"

```
# CMakeLists.txt for bar project
find_package(foo REQUIRED)
add_executable(bar bar.cpp)
target_link_libraries(bar foo)
```

"c++ -std=c++11 … /path/to/libfoo.so"

Propagation of Compile Features

CMake and Other Documentation

- Compile features have a dedicated section in CMake’s documentation:
  - https://cmake.org/cmake/help/v3.3/manual/cmake-compile-features.7.html
- C++ Support Status for GNU, Clang, Intel and Microsoft compilers
- C++ Standard Libraries, GNU, LLVM, Microsoft