Report from Parallel 6B: C++11 Migration

Ben Morgan

THE UNIVERSITY OF WARWICK

Overview

- “C++11 Compilers Support”: Andrea Dotti
  - *Versions for 10.2 to support C++11, How to Install*
- “CMake Support for C++11/14/1z”: Ben Morgan
  - *Checks/workarounds for C++11 and beyond*
- “Mini Course on C++11 Features”: Ivana Hrivnacova
  - *Guidelines for using C++11 in Geant4*

C++11

SLAC

Platforms we plan to support for version 10.2 are:

- OS:
  - SLC6 with latest compiler
  - Linux CentOS-7 (coming with gcc-4.8.2 vanilla)
  - MacOS Yosemite
  - Window 7 or 8 (or 10)
- Compilers:
  - **gcc-4.8.1 or greater**
  - **clang-3.5 or greater**
  - **icc-15 or greater**
  - **Visual-C++ 14 (Visual Studio 2015)**

El Capitan builds likely, but maybe not fully tested by release deadline

Guide on installing and using gcc on older systems, Intel/Clang with modern libstdc++

Geant4 10.2 will automatically recognize the compiler features and add the appropriate flags at compile time (no more need for cmake GEANT4_BUILD_CXXSTD=c++11 flag)

4

C++11 Compiler Support

ICC and Xeon Phi

SLAC

We have identified an issue with icc (even in the very latest versions) when compiling Geant4 with c++11 for Xeon Phi

- due to missing full support of thread_local storage for non-POD types

Acknowledged by Intel developers, should be fixed in next mpss stack update (tba):

- follow updates at: https://software.intel.com/en-us/forums/intel-c-compiler/topic/593648
- Gabriele provided a workaround in external category, final fix planned for 10.2 after Intel feedback

C++11 Compiler Support: Xeon Phi

```
# 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
  )
```

CMake extension to check and validate C++ Standard Library Features

CMake Builtin to assert that target requires compiler support for C++11/14 syntax features

Checking C++ Library/Language Features

CMake Support for C++11/14/1z

```
// - foo.cpp
#include “foo_compiler_s
#include “foo_stdlib_sup
…
CCF_THREAD_LOCAL myTLVariable;

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

Builtin Support to workaround missing features by auto generated preprocessor macros

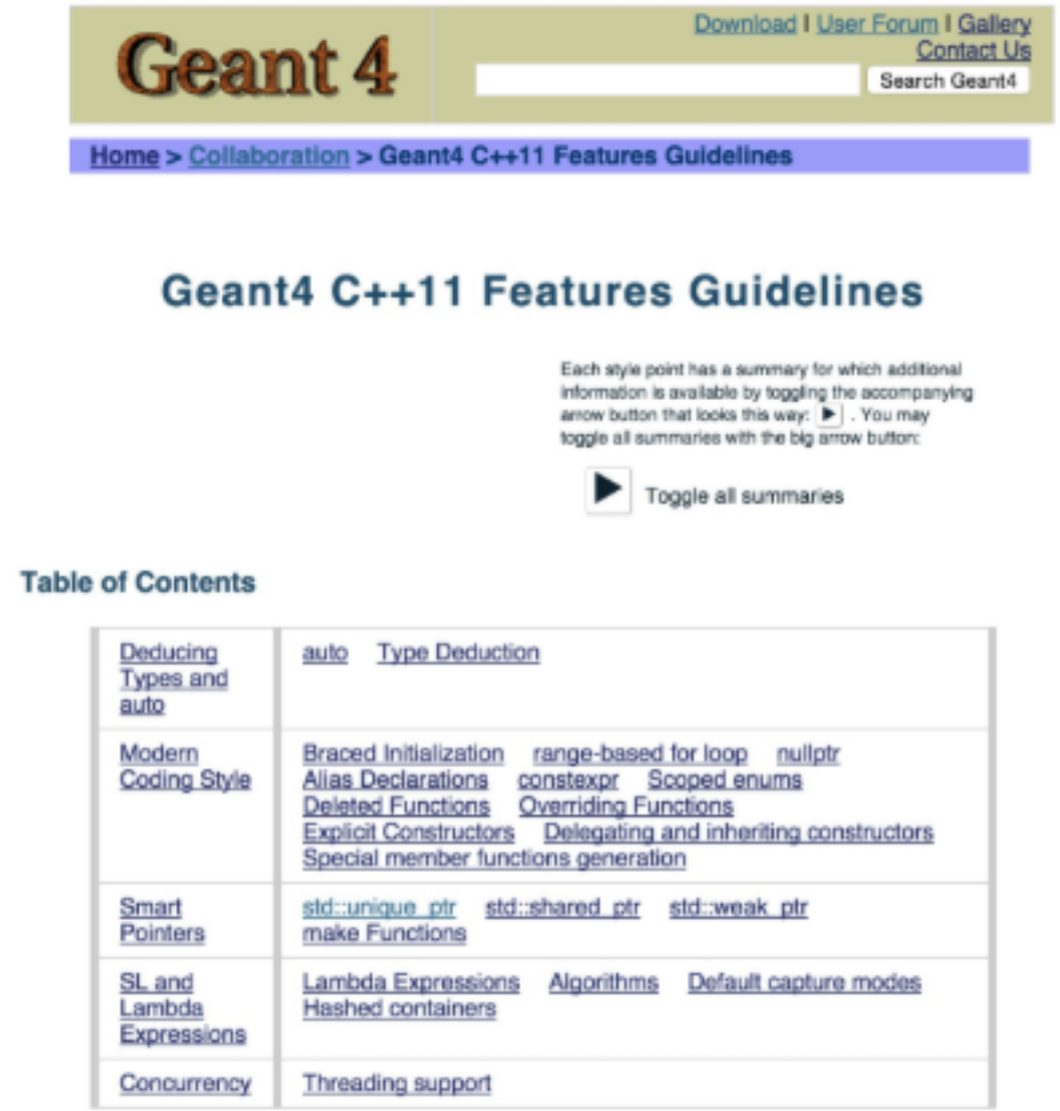
Extensions to provide implementations of C++<NEXT> features in C++<NOW>

Working Around Missing Features: Code

CMake Support for C++11/14/1z

C++11 Guidelines Document

- Compiled from the following sources:
  - Effective Modern C++ by Scot Meyers (O'Reilly). Copyright 2015 Scot Meyers. 978-1-491-90399-9.
  - ALICE O² C++ Style Guide.
  - cplusplus.com
  - Stack Overflow
  - *It is using style sheets of C++ Google Style guide, Revision 3.274 (link) under the CC-By 3.0 License further modified by ALICE O² project*



Geant 4

Download | User Forum | Gallery
Contact Us
Search Geant4

Home > Collaboration > Geant4 C++11 Features Guidelines

Geant4 C++11 Features Guidelines

Each style point has a summary for which additional information is available by toggling the accompanying arrow button that looks this way: ▶ . You may toggle all summaries with the big arrow button:

▶ Toggle all summaries

Table of Contents

Deducing Types and auto	auto Type Deduction
Modern Coding Style	Braced Initialization range-based for loop nullptr Alias Declarations constexpr Scoped enums Deleted Functions Overriding Functions Explicit Constructors Delegating and inheriting constructors Special member functions generation
Smart Pointers	std::unique_ptr std::shared_ptr std::weak_ptr make Functions
SL and Lambda Expressions	Lambda Expressions Algorithms Default capture modes Hashed containers
Concurrency	Threading support

http://geant4.cern.ch/collaboration/c++11_guide.shtml

Mini Course on C++11 Features

To help Geant4 developers use/migrate C++11 features

Extra info, examples and rationale on guidelines, plus move semantics Will be uploaded as accompanying material to Guidelines Page

auto (2)

- Auto variables are immune to type mismatches

```
std::unordered_map<std::string, int> m;
for (const std::pair<std::string, int>& p : m) {
  ...   // do something with p
}
```

  - The p type in the loop does not match the map m element type, which is **std::pair<const std::string, int>** (note the const)

```
std::unordered_map<std::string, int> m;
for (const auto& p : m) {
  ...   // do something with p
}
```


Mini Course on C++11 Features

Summary

- Geant4 ready to move forward with C++11

- Get your development toolkit set up using the info in Andrea's presentation

- Get coding with C++11 following Guidelines and Ivana's course presentation