C++11 Compilers support

A. Dotti ; SLAC SD/EPP/Computing

Overview

SLAC

c++11 is a new standard of c++ language, it includes new features and extensions to the language (e.g. move semantic, curly-brace initialization, lambdas) as well as new algorithms and containers in the standard template library (e.g. std::unoredered_map, std::shared_ptr, std::thread)

c++11 is also faster, because of such new features as move semantics and because it increases the opportunities for compiler optimizations

Code is obviously not back-compatible, but pre-c++11 code compiles with modern compilers: you need a relatively recent version of a compiler to enable these features (usually w/ flag –std=c++11)

Geant4 and c++11/14

SLAC

Geant4 Version 10.2 will be released with c++11 native code

- it will compile only with new compilers
- maintainence patches to 10.1 and 10.0 will be backported to pre-c++11

We are also interested into c++14, but this is a minor update of the standard (put in place what did not manage in time for c++11: e.g. literals)

Some features are extremely valuable for Geant4: std::thread, thread_local

- allow MT on WIN (?)

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)**

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)

How to pre-installed compilers

SLAC

The easier way to get on Linux the latest compiler is to move the OS on CentOS 7 (or similar), no further action required…

- to ease the transition, you can test a VM, for example the one provided by our IN2P3 colleagues: http://geant4.in2p3.fr/spip.php?rubrique8

At CERN on lxplus (SLC6), you can setup a recent compiler via e.g.:
source /afs/cern.ch/sw/lcg/external/gcc/4.9.2/x86_64-slc6/setup.sh

Intel Compiler is also available:

- source /afs/cern.ch/sw/IntelSoftware/linux/setup.sh (to setup license)
- source /afs/cern.ch/sw/IntelSoftware/linux/x86_64/xe2016/bin/compilervars.sh intel64

How to install gcc

SLAC

Even on old (linux) systems installing a recent gcc is not difficult (but long): https://gcc.gnu.org/install/

Following example to install gcc 5.2 for linux box where a system gcc is already available: absolute minimum version is 3.4, but strongly suggest at least 4.8. Better to bootstrap from recent version, do a multi-version installation for extra-security, first install gcc-4.8 then others

1. Create an emtpy dir where install sources, e.g. `mkdir ~/gcc-src && cd ~/gcc-src`
2. Download pre-requisites:
   a) gmp library (GNU Multiprecision Library) >= 4.3.2 : https://gmplib.org/
   b) mpfr library (Multi Precision Floating-point computations with correct Rounding) >= 2.4.2 : http://www.mpfr.org/
   c) mpc library (Multi Precision C) >= 0.8.1 : http://www.multiprecision.org/

How to install gcc (2)

SLAC

3. Download gcc source from one of the mirrors () , e.g.:
   a) `wget http://www.netgull.com/gcc/releases/gcc-5.2.0/gcc-5.2.0.tar.gz`
   b) `wget http://www.netgull.com/gcc/releases/gcc-5.2.0/md5.sum`
   c) `md5sum gcc-5.2.0.tar.gz (and check checksum with content of md5.sum)`
   d) `tar xzf gcc-5.2.0.tar.gz`
4. Un-tar pre-requisites into gcc source directory, removing version number:
   a) `tar xjf mpfr-*.tar.bz2 && mv mpfr-* gcc-5.2.0/mpfr`
   b) `tar xjf gmp-*.tar.bz2 && mv gmp-* gcc-5.2.0/gmp`
   c) `tar xzf mpc-*.tar.gz && mv mpc-* gcc-5.2.0/mpc`

How to install gcc (3)

SLAC

5. Configure gcc, out-of-source build:
   a) `mkdir ~/gcc-src/build && cd ~/gcc-src/build`
   b) `./configure –prefix=<some-directory> --enable-languages=c,c++`

   Note: installing gcc in alternative directory under: <some-directory> (e.g. /usr/local/gcc-5.2), c and c++ are the minimum for Geant4, you may want to enable also fortran
6. Compile, via bootstrap. gcc will be compiled three times (all automatic, you do not need to care): first time use system compiler, then used freshly build compiler to build itself, repeat a third time to check that stage 2 and stage 3 give identical results:
   a) `make –j <N>`
   b) `Optional: make –k check (requires DejaGNU, TCL and Expect)`
   c) `make install`
7. Go for lunch…

How to use alternative gcc

SLAC

To use laternative gcc (installed in /opt/gcc-5.2) you need to use the following setup:

```
basedir=/opt/gcc-5.2
export PATH=${basedir}/bin:$PATH
export COMPILER_PATH=${basedir}/libexec/gcc/x86_64-
unknown-linux-gnu/5.2.0
export LD_LIBRARY_PATH=${basedir}/lib64:$LD_LIBRARY_PATH
export LD_RUN_PATH=${basedir}/lib64:$LD_RUN_PATH
```

Configure Geant4 with:

```
export CC=gcc
export CXX=g++
cmake […]
```

Non gcc compiler

SLAC

I've created a twiki describing issues for c++11 and Geant4:

- https://twiki.cern.ch/twiki/bin/view/Geant4/Cxx11MigrationTaskForce
- feel free to contribute

Some notes:

- Mac OS X: use clang, alternative gcc installation is very painful and did not succeed (see twiki)

If you want (on linux) to use an alternative (clang or icc) compiler you should note the following important notes:

- you still need a recent gcc compiler installed and configured (both clang and icc use the system libstdc++ and headers file that must be c++11 compatible)
- with icc you cannot use a too new version of gcc: stick to a 4.9.x

Non gcc compiler on linux

SLAC

If you want to use clang:

1. `source /opt/gcc-4.9.x/setup.sh`
2. `export CC=clang`
3. `export CXX=clang++`
4. `cmake -DCMAKE_CXX_FLAGS="--gcc-toolchain=/opt/gcc-4.9.x" […]`

For intel compiler it is enough that the alternative gcc is in path:

1. `source /opt/gcc-4.9.x/setup.sh`
2. `export CC=icc`
3. `export CXX=icpc`
4. `cmake […]`

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

Conclusions

SLAC

Geant4 w/ c++11 native code, compiles on all supported platforms

For Linux stick to gcc, for Mac stick to clang

- installation of alternative (recent) gcc is relatively simple

Xeon Phi compilation has posed some challenges, currently investigating

- provided feedback to Intel developers and their help is very appreciated