HepMC

Lynn Garren

June 15, 2011

HepMC 2.07

- There has long been a problem with HepMC and root I/O
- The experiments all have their own solutions
- Geant effort would like a real solution
- A few other miscellaneous items

HepMC and ROOT

- Pointers make it difficult for root to use advance I/O features
  - back pointers especially problematic
- root likes to work in chunks
  - HepMC GenEvent must be dealt with as a whole
  - one chunk might be a list of GenParticles

HepMC and ROOT

DON'T PANIC

stay calm and go to the pub

Constraints

- Backwards compatibility
- Backwards compatibility
- Backwards compatibility
- existing code MUST continue to work
- existing files MUST continue to work

Transient vs Persistent

- Pointers, especially backward pointers, are problematic
- std::map also a problem for root
- ASCII I/O treats pointers as transient and recreates them
- Leverage this for root
  - persist information as integers
  - barcode not going to work for this

Considerations

- Possible to create transient data on the fly
  - impacts compute time
  - recall that HepMC is designed to supply pointers
- Users may add vertices and particles to the event
- Create all transient data when building GenEvent

GenEvent container

- replace map with a list or vector
- vector of objects
  - if you add or remove objects, the vector members may be rearranged in storage, making pointers invalid
- vector of pointers
- list of objects
  - adding or removing member does not change where other members are stored

Container choice

- std::vector<GenVertex*>
  - HepMC designed with the idea that users will be working with pointers to GenEvent, GenVertex, Genparticle
- std::list<GenVertex>
  - Ownership is explicit
- opt for std::vector

The Plan

- Explicitly identify persistent and transient data for each class
- Replace maps with vectors
- Allow for the possiblity that a GenVertex or GenParticle may be detached from its GenEvent
  - recall that root will do this

GenEvent transient data

- GenVertex* signal_process_vertex
- GenParticle* beamparticle_1
- GenParticle* beamparticle_2
- std::map< int,HepMC::GenVertex*,std::greater<int> > vertex_barcodes
- std::map< int,HepMC::GenParticle*,std::less<int> > particle_barcodes

GenEvent persistent data

- int signal_process_id
- int event_number
- int mpi
- double event_scale
- double alphaQCD
- double alphaQED
- WeightContainer weights
- std::vector<long> randostates
- std::vector<HepMC::GenVertex*> vertices NEW
- std::vector<HepMC::GenParticle*> particles NEW
- GenCrossSection* cross_section
- HeavyIon* heavy_ion
- PdfInfo* pdf_info
- Units::MomentumUnit momentuunit
- Units::LengthUnit position_unit

GenVertex transient data

- std::vector<HepMC::GenParticle*> particles_in
- std::vector<HepMC::GenParticle*> particles_out
- GenEvent* event

GenVertex persistent data

- FourVector position
- std::vector<size_t> particles_in_index NEW
- std::vector<size_t> particles_out_index NEW
- int id
- WeightContainer weights
- WeightContainer weights
- int barcode

GenParticle transient data

- GenVertex* production_vertex
- GenVertex* end_vertex
- GenEvent* parent_event NEW
  - required by the idea of detached particle

GenParticle persistent data

- FourVector momentum
- int pdg_id
- int status
- Flow flow
- Polarization polarization
- size_t production_vertex_index NEW
- size_t end_vertex_index NEW
- int barcode
- double generated_mass

IO_MockRoot

- proof of principle for root data storage scheme
- not meant to replace IO_GenEvent
- also provide linkdef file

IO_GenEvent format

E - general GenEvent information

N - named weights

U - momentum and position units

C - GenCrossSection information (This line will appear ONLY if GenCrossSection is defined.)

H - HeavyIon information (This line will appear ONLY if HeavyIon defined.)

F - PdfInfo information (This line will appear ONLY if PdfInfo defined.)

V - GenVertex information

P - GenParticle information

P - GenParticle information

V - GenVertex information

P - GenParticle information

P - GenParticle information

P - GenParticle information

IO_MockRoot format

E - general GenEvent information

N - named weights

U - momentum and position units

C - GenCrossSection information (This line will appear ONLY if GenCrossSection is defined.)

H - HeavyIon information (This line will appear ONLY if HeavyIon defined.)

F - PdfInfo information (This line will appear ONLY if PdfInfo defined.)

V - GenVertex information

V - GenVertex information

V - GenVertex information

P - GenParticle information

P - GenParticle information

P - GenParticle information

Rearrange examples

- Many existing examples rely on external packages
- HepMC itself has NO dependencies
- move all examples which depend on external packages to appropriate subdirectories
- HepMC 2.06 and 2.07

New example directory

- example_EventSelection.cc example_UsingIterators.cc
- clhep
  - example_BuildEventFromScratch.cc
- herwig
  - example_MyHerwig.cc testHerwigCopies.cc
- pythia
  - example_MyPythia.cc example_MyPythiaOnlyToHepMC.cc
    example_PythiaStreamIO.cc testPythiaCopies.cc

libtool again

- Libtool recognizes and deals with many different compilers, but sometimes uses options that cause compilation problems.
  - libtool is embedding full paths in MacOSX shared libraries.
  - At that level, libtool seems to ignore directives passed to it via autoconf/automake.
- Already not possible to use libtool for VC++.
- drop libtool?
- support cmake as an alternate build method

Future

- Experiments to move from 2.03 to 2.06
  - WHEN????
- Drop support for 2.03 at the end of 2011
- Windows XP becoming obsolete
  - support to be reevaluated at end of 2011
- prefer not to build 2.07 for Windows