HepMC

Lynn Garren

February 3, 2010

Where to find HepMC

- Homepage:
- Bug Reports:
- Downloads:
- LCG builds:

Supported Platforms

osx105_ia32_gcc401	
i386-mac106-gcc42-opt	x86_64-mac106-gcc42-opt
slc4_amd64_gcc34	slc4_amd64_gcc43
slc4_ia32_gcc34	slc4_ia32_gcc43
i686-slc5-gcc34-opt	
i686-slc5-gcc41-opt	i686-slc5-gcc43-opt
x86_64-slc5-gcc34-opt	
x86_64-slc5-gcc41-opt	x86_64-slc5-gcc43-opt
win32-vc71	i686-winxp-vc9-opt

Current HepMC Releases

- 2.05.01 - Jan. 6, 2010
- 2.03.11 - June 3, 2009
- proposals for 2.06.00
  - drop support for 2.05 when 2.06 is released?

Proposals for 2.06

- allow named weights in WeightContainer
- iterator options
- I/O requests
- remove deprecated classes
- Polarization member of GenParticle
- optional function to reduce HepMC size
- Pythia and Herwig wrappers
- installed location of examples

Named Weights 1

- weights have been used to deal with different PDFs
- attaching names to the weights would make their use more transparent
- standardized names would be nice but they are not an option at this time
- index weights with arbitrary std::string
- cannot change existing functionality

Named Weights 2

- GenEvent methods:
    - WeightContainer&   weights();
    - const WeightContainer& weights() const;
- WeightContainer contains vector<double>
- add an internal map of strings to vector index
- new methods hide internal map to mimic map of strings to double
- all existing WeightContainer methods unmodified

Named Weights 3

- double& operator[](const std::string&);
- const double& operator[](const std::string&) const;
- bool operator==(const WeightContainer &) const;
- bool operator!=(const WeightContainer &) const;
- bool search(const std::string& s) const;
- WeightContainer w;
- w["myPDF"] = 2.3425;
- default weight names are the vector index

Iterator Options

- add container-like access to iterators for convenience
    - this could be a huge win for users
- original request for, e.g., vector<GenParticle*>
    - iterate just to produce the vector
    - this copy is not connected to the event
- can provide this functionality a better way

Existing Public Iterators

- GenEvent::vertex_const_iterator
- GenEvent::vertex_iterator
- GenEvent::particle_const_iterator
- GenEvent::particle_iterator
- GenVertex::vertex_iterator
- GenVertex::particle_iterator

GenEventIterators.h

- new classes that look like a container to STL and Boost methods:
  - GenEventParticleIterator(GenEvent &);
  - ConstGenEventParticleIterator(GenEvent const &);
  - GenEventVertexIterator(GenEvent &);
  - ConstGenEventVertexIterator(GenEvent const &);
  - GenVertexParticleIterator(GenVertex &, IteratorRange r = relatives);
  - GenParticleProductionIterator(GenParticle const &, IteratorRange r = relatives);
  - GenParticleEndIterator(GenParticle const &, IteratorRange r = relatives)
- all return the appropriate begin() and end() iterators

existing code

```
for ( HepMC::GenEvent::particle_iterator p = evt->particles_begin();
       p != evt->particles_end(); ++p ) {
  if ( (*p)->end_vertex() ) {
    for ( HepMC::GenVertex::particle_iterator d
      =(*p)->end_vertex()->particles_begin(HepMC::descendants)
      d != (*p)->end_vertex()->particles_end(HepMC::descendants);
               ++d )
         {  (*d)->something();   }
       } // loop over descendants
  } // if end vertex
}  // all GenEvent particles
```

new code

```
HepMC::GenEvent* evt;
HepMC::GenEventParticleIterator pc(*evt);
for ( HepMC::GenEvent::particle_iterator p = pc.begin();
          p != pc.end(); ++p ) {
      if ( (*p)->end_vertex() ) {
        HepMC::GenParticleEndIterator d(*p,HepMC::descendants);
         std::for_each( d.begin(), d.end(), UseDescendants());
        // loop over descendants using std::for_each
         }  // if end vertex
} // loop over all GenEvent particles
```

Iterator Comments

- once you have begin() and end() iterators, it looks like a collection
- there are no const equivalents for the GenVector iterators (taking a range)
- GenParticleProductionIterator throws if production_vertex is undefined
- GenParticleEndIterator throws if end_vertex is undefined
- GenVertexVertexIterator not implemented

ASCII I/O Request

- only write HeavyIon and PdfInfo lines in ASCII output if present
  - requires backporting
- allow random lines to be inserted in the header portion of ASCII output
  - first character cannot collide with defined line keys (E, N, U, C, H, F, V, P)
- vertex and particle section of GenEvent will NOT support this

Binary I/O

- various linear collider, etc. users are requesting binary I/O
- Pere suggests adding a new library for binary ROOT I/O
- Norman Graf explicitly requests XDR I/O
- NEITHER should be built by default

Binary I/O Suggestion

- by long standing agreement, HepMC has no dependencies
- that cannot change
- both binary I/O options can be made available in some parallel fashion
- should such source code be packaged with the main HepMC source code?

Binary I/O Package Options

- put HepMC ROOT IO in ROOT
  - reject
- add directories in HepMC
  - rootIO and xdrIO
  - possible dependency confusion
  - must explicitly ask to build these
- parallel packages
  - HepMCrootIO and HepMCxdrIO

Remove Deprecated Classes

- remove deprecated ParticleDataTable, ParticleData, and IO_PDG_ParticleDataTable
- changes in IO_BaseClass
- coded in HEPMC_02_06_proposal
- proposed for inclusion at July meeting

GenParticle Polarization

- there is no way to specify that a GenParticle Polarization is undefined
- problematic, e.g., for tau physics
- proposal: store a pointer to Polarization
  - not backwards compatible

GenParticle Polarization

- methods
  - const Polarization & polarization() const;
  - void set_polarization(const Polarization& pol = Polarization(0,0));
- data member
  - Polarization m_polarization;
- alternative: is_set in Polarization??

Reduce HepMC Size

- provide method(s) to remove particles which do not have status 1, 2, or beam
- Andy has written an external function that does this
- reduces volume of data passed between generator and analysis
- possible loss of interesting information
- intended as a testing tool

A Caution

- If we want one simple option, relevant e.g. for Rivet analyses of the event record, it would be one where ONLY status code 1 particles are retained, i.e. the final ones. This is a clean concept, with no ambition whatsoever to retain history information.
- There is then no memory of intermediate B's or taus, except by checking for particles with a common displaced vertex. Retaining status 2 keeps the intermediate hadrons.
- But if, say, you want to relate the partonic activity to final jets, you need the whole event. (Assuming status codes 11 - 200 are used to trace shower evolution and the like.)
- Such a method has specialized utility, but is probably not something that should be in general use.

Pythia & Herwig Wrappers

- provide a single Pythia wrapper
  - based on Pythia 6.4
  - PythiaWrapper.h
- provide a single Herwig wrapper
  - based on Herwig 6.5
  - HerwigWrapper.h
- make sure these fill all relevant info
- add some useful methods

install locations

- Use datarootdir and follow general usage for example and document directories
- i686-slc5-gcc43-opt/ include lib (unchanged)
- i686-slc5-gcc43-opt/ doc examples (move)
- new structure:
  - i686-slc5-gcc43-opt/share/ doc examples
- --datarootdir=DIR read-only arch.-independent data root [PREFIX/share]
- --docdir=DIR documentation root [DATAROOTDIR/doc/HepMC]

const method reminder

- Methods modified as of 2.04 to return a const reference instead of a copy
- classX method() const;

  ==> const classX & method() const;

  - GenParticle::flow()
  - GenParticle::polarization()
  - GenEvent::random_states()
- not completely backwards compatible
  - but definitely consistent with code intent
- some client code may need to be modified