Recent developments in the HepMC event record

http://hepmc.web.cern.ch/hepmc/

hepmc-dev@cern.ch

Andrii Verbytskyi[a], for the HepMC authors [1]

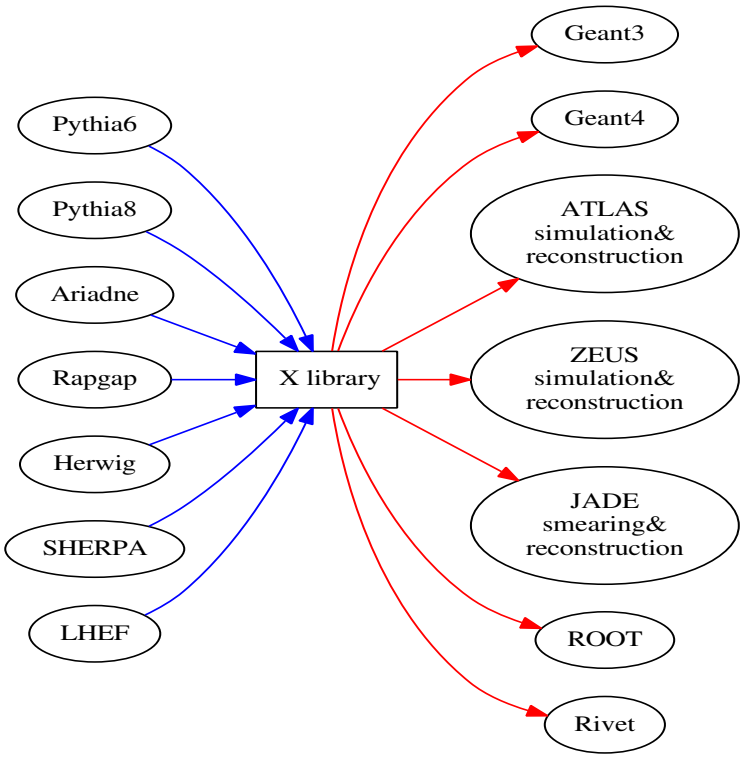
[a] Max-Planck-Institut für Physik (Werner-Heisenberg-Institut)

WLCG/HSF Workshop at DESY Hamburg, May 13-17 2024, Hamburg, Germany

What is event record

- Event record contains all physical information on the initial, intermediate and final state particles in the simulated collision: particle flavours, momenta, production coordinates, etc.
- A standardised event record is essential for the HEP/particle physics experiment, theory and phenomenology.

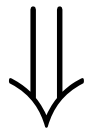
Expectation from a library for event record



- The typical task is to pass MC generated events between different generators, reconstruction programs and/or analysis frameworks.
- $\rightarrow$ **No universal connector library existed before 201X!**
- Before the HepMC3-3.0.0 the closest match was de-facto a standard HepMC2 [2] and a beta version of HepMC3.

What is HepMC3: Definition and application

- HepMC3 is a library designed to operate with Monte Carlo event records in High Energy physics (HEP), see Ref. [1].



- The library should be used to store and transfer Monte Carlo event records between different HEP software and/or disk.
- HEP software in focus: Monte Carlo event generators (e.g. Pythia8[3], Herwig7 [4], SHERPA [5]), simulation software (e.g. Geant4 [6], FLUKA [7]), analysis frameworks (ROOT [8], experiment-specific), plotting tools (e.g. Rivet [9]) etc.

People, projects and principles

People and projects

- Code contributors: T. Przedzinski, AV, A. Buckley, W. Pokorski, M. Ellert, D. Konstantinov, L. Lönnblad, J. Monk, M. Kirsanov, L. Pickering, H. Schulz, P. Onyisi, A.P. Neuwirth, A. Kabelschacht, N. Dawe, R. Isemann, P.J. Ilten, A. Sailer, S. Snyder
- Issue reporters: J. Whitehead, J. Holeczek, S.R. Klein, C.H. Christensen, L.J. Nevay, S. Banerjee, F. Siegert, C. Gutschow, T. Mombaecher, H. Dembinski, P. Mato Vila, M. Elashri, P. Demin, P. Sznajder, E. Olatunji Olaiya, N. Hartmann, K. Pedro, (E. Botmann, J. Chapman and others via e-mails)
- Many thanks to the related projects:
  - Python C++ bindings ‘pybind11‘ [10]
  - Python bindings generator ‘binder‘ [11]
  - Decompression libraries interface ‘bxzstr‘ [12]
- Related projects: ‘pyhepmc‘ [13] and ‘NuHepMC‘ [14].
- Packaging: M. Ellert, F. Siegert, E. Botmann, A. Bhattacharya, AV.

Development principles

- The core library should not have dependencies but C++11 STL.
- The optional submodules should be kept well separated.
- Backward compatibility of serialisation formats.
- API changes are not desirable.

Typically almost no client code updates are needed to upgrade to the next HepMC3 version.

Timeline of HepMC3

- Mar 20, 2014: Initial commit by Tomasz Przedzinski – a project together with Witold.
- Jul 13, 2014: Andy Buckley joins.
- 2016: Beta version.
  Renaming the namespace from HepMC:: to HepMC3::, Cancellation of the HepMC v 2.07. Port to Windows. Proper ROOT support.
- Mar 2017: 3.0.0
  Preparation of the HepMC3 paper. Packaging for Fedora, EPEL and Ubuntu. Development of the 'search' library.
- Feb 2019: 3.1.0
  Python interface and a parallel project 'pyhepmc' by Hans Dembinski. Adoption into LCG stack.
- Nov 2019: 3.2.0
  Adoption by various MCEGS, ATLAS, removal of MCEG interfaces from the HepMC3 codebase. Support of Protobuf I/O. Removal of the old CERN JIRA HepMC ticket system. Support of compression I/O. ...
- Oct 2023: 3.2.6
  NuHepMC
- Oct 2023: 3.2.7
  Rivet 4 relies on HepMC3. Drop Python2 support.
- April/May 2024: 3.3.0?
- Future plans

Current status

Current usage

- As of 2024 most widely used collider MCEGs use HepMC3 either as the only event record library or as an alternative to HepMC.
- ATLAS completed software stack transition to HepMC3 in 2023. Other LHC experiments use HepMC3 either directly or indirectly.
- Significant interest in Heavy Ion MCEG community (e.g. SMASH) and neutrino physics community (NuHepMC).
- (Official) packages exist for Fedora, EPEL, OpenSuse, ArchLinux, homebrew(Mac OS X), PyPi(Windows,Linux,Mac OS X). **That is the recommended way of installation**. Compilation from sources is possible on multiple platforms.

Support, optimisation, bug fixes

Example: preparations for 3.3.0

≈ 100 merge requests

- Dropped Python2 support and related Cmake scripts
- Updated the documentation format to MD
- Added ‘deduce_reader‘ to python bindings
- Updates of the embedded code
- Bugfixes
  - https://gitlab.cern.ch/hepmc/HepMC3/-/merge_requests/338
  - https://gitlab.cern.ch/hepmc/HepMC3/-/merge_requests/341
  - https://gitlab.cern.ch/hepmc/HepMC3/-/merge_requests/318
  - https://gitlab.cern.ch/hepmc/HepMC3/-/merge_requests/315
  - https://gitlab.cern.ch/hepmc/HepMC3/-/merge_requests/297
  - https://gitlab.cern.ch/hepmc/HepMC3/-/merge_requests/340
- API changes
  - e.g. https://gitlab.cern.ch/hepmc/HepMC3/-/merge_requests/342

Requirements for HepMC3 3.3.0

Minimal:

- Modern Linux, OSX, BSD, Solaris or Windows system
- C++11 compiler (gcc/clang/Intel/Oracle SunPro/PGI/IBM XL)
- CMake 3.7+

Recommended=Minimal+

- ROOT6
- protobuf library and compiler

Full=Recommended+

- doxygen, latex, graphviz for documentation
- valgrind, Pythia8, HepMC2 for tests
- Fortran77 compiler for examples
- Python3/PyPy modules

Recent developments in code

protobuf I/O

The support for the protobuf I/O was gradually added in the 3.2.X branch.

- Required by NuHepMC.

Python bindings

The Python bindings were added to HepMC3 in version 3.2.0. As of 3.3.0 the bindings

- can be compiled with the CPython 3.6 or with PyPy.
- cover the whole codebase and have syntax identical to C++ functions.
- are generated automatically using ‘binder‘ [11] and require for the compilation the ‘pybind11‘ [10] header-only library embedded in the HepMC3 code.

An alternative set of python bindings is provided by the ‘pyhepmc‘ [13] project.

Support of the compressed I/O

The support for the compressed I/O was gradually added in the 3.2.X branch.

- It is implemented via 'ReaderGZ/WriterGZ' classes which wrap the normal readers/writers.
- The 'ReaderGZ/WriterGZ' classes compress/decompress the I/O streams using the 'bxzstr' [12] header-only library embedded in the HepMC3 code.
- The code should be linked to any of 'zlib', 'bz2', 'lzma' or 'zstd' libraries.
- The python bindings can use the standard python libraries for decompression. WIP.

I/O utilities

- HepMC3::deduce_reader is designed to guess the input format of the given stream or file. Has a python version.
- HepMC3::ReaderMT is designed to read the input file in multiple streams using multithreading.
- HEPEVT_Wrapper_Runtime is designed to be able have different sizes of HEPEVT block.
- ReaderPlugin/WriterPlugin – plugin loading mechanism.

Ideas for the future

HepMC3 for Heavy Ion

- GenHeavyIon is a standard attribute to hold information in Heavy Ion collisions, significantly expanded in comparison to HepMC2. Close to Lisbon Accord [15], but can be extended even more to fulfil needs of Heavy Ion experiments.
- Support of Readers for e.g. OSCAR format?
- Speedup the I/O in the default readers.
- Extending Heavy Ion event record standard.

```
1   int    Ncoll_hard;                      /// the number of hard nucleon-nucleon collisions.
    int    Npart_proj;                      /// the number of participating nucleons in the projectile.
3   int    Npart_targ;                      /// the number of participating nucleons in the target.
    int    Ncoll;                           /// the number of inelastic nucleon-nucleon collisions.
5   int    N_Nwounded_collisions;           /// Collisions with a diffractively excited target nucleon.
    int    Nwounded_N_collisions;           /// Collisions with a diffractively excited projectile nucleon.
7   int    Nwounded_Nwounded_collisions;    /// Non-diffractive or doubly diffractive collisions.
    double impact_parameter;                /// The impact parameter.
9   double event_plane_angle;               /// The event plane angle.
    double sigma_inel_NN;                   /// The assumed inelastic nucleon-nucleon cross section
11  double centrality;                      /// The centrality.
    double user_cent_estimate;              /// A user defined centrality estimator.
13  int Nspec_proj_n;                       /// The number of spectator neutrons in the projectile
    int Nspec_targ_n;                       /// The number of spectator neutrons in the target
15  int Nspec_proj_p;                       /// The number of spectator protons in the projectile
    int Nspec_targ_p;                       /// The number of spectator protons in the target
17  map<int,double> participant_plane_angles; /// Participant plane angles
    map<int,double> eccentricities;         /// Eccentricities
```

More information for specific event types

- Add a module with extra attributes, e.g. to hold information on TMDs, Heavy Ion observables, DVCS observables, polarisation of beams, etc.

HDF5

- Recently HDF5 format became popular among the MCEG developers.
- Essentially that is LHE events serialised with HDF5.
- Possible extension for the LHEF in HepMC3.

Fast and bulk I/O

- ROOT bulk I/O can be an interesting option for phenomenology.
- The python bindings can be extended with options for pandas [16] I/O.
- Support of writing with ‘uproot‘ [17] (reading is implemented).

Introduce plugin I/O into the interfaces of MCEGs and tools

- Some users would prefer their own output formats, i.e. own way to serialise HepMC::GenEvent.
- The HepMC3 plugin mechanism can be used:
  - The plugin should be compiled by user into a library.
  - A MCEG has to be provided with the location of the library.

Similar mechanisms can be implemented into the tools that consume HepMC, e.g. Rivet.

Summary

- Collecting feedback from LHC experiments, MCEG authors and interested individuals.
- If you have a good idea, feature request or bug report, please, create an issue in https://gitlab.cern.ch/hepmc/HepMC3 or send a mail to hepmc-devATcern.ch mailing list. **We do answer mails!** .

Visit http://hepmc.web.cern.ch/hepmc/

Write an e-mail to hepmc-dev@cern.ch or to any of authors.

Backup slides

Example of event analysis in ROOT (Headers omitted).

```
class SomeAnalysis{
public :
    TChain          *fChain;   //!pointer to the analyzed TTree or TChain
    Int_t           event_number;
    Int_t           momentum_unit;
    TBranch         *b_hepmc3_event_event_number;   //!
    TBranch         *b_hepmc3_event_momentum_unit;   //!
    TBranch         *b_hepmc3_event_length_unit;   //!
    TBranch         *b_hepmc3_event_particles_;   //!
    void Init(TChain *tree) {
        if (!tree) return;
        fChain = tree;
        fChain->SetMakeClass(1);
        fChain->SetBranchAddress("event_number", &event_number, &b_hepmc3_event_event_number);
    SomeAnalysis(const std::string& file) {
        TChain* TempChain= new TChain("hepmc3_tree");
        TempChain->Add(file.c_str());
        Init(TempChain);
    }
};
int main(){
    TH1D H1("H1","Pt of pions or electrons;Events/100MeV;P_{T},GeV",1000,0,100);
    SomeAnalysis* A= new SomeAnalysis("inputIO4.root");
    if (!A->fChain->GetEntries()) return 1;
    for (int entry=0; entry<A->fChain->GetEntries(); entry++){
            A->fChain->GetEntry(entry);
            for (int i=0; i<A->particles_; i++)
                if (A->particles_status[i]==1&&(std::abs(A->particles_pid[i])==211||std::abs(A->particles_pid[i])==11))
                    H1.Fill(std::sqrt(A->particles_momentum_m_v1[i]*A->particles_momentum_m_v1[i]+A->particles_momentum_m_v2[i]*
        A->particles_momentum_m_v2[i]) );
    }
    delete A;
    H1.Print("All");
    return 0;
}
```

- Reading file in ROOT format without HepMC3.

Event analysis listing of ROOT HepMC3 tree, with HepMC3

```
#include "HepMC3/GenEvent.h"
#include "HepMC3/GenParticle.h"
#include "HepMC3/ReaderRootTree.h"
#include <TH1D.h>
int main()
{

    TH1D H2("H2","Pt of pions or electrons;Events/100MeV;P_{T},GeV",1000,0,100);
    HepMC3::ReaderRootTree inputA("inputIO4.root");
    if(inputA.failed()) return 10002;
    while( !inputA.failed() )
        {
            HepMC3::GenEvent evt(HepMC3::Units::GEV,HepMC3::Units::MM);
            inputA.read_event(evt);
            if( inputA.failed() )  {printf("End of file reached. Exit.\n"); break;}
            for(auto p: evt.particles() )
            if ( std::abs(p->status()) == 1 && (std::abs(p->pdg_id()) == 211||std::abs(p->pdg_id()) == 11) )
                H2.Fill( p->momentum().perp());
            evt.clear();
        }
    inputA.close();
    H2.Print("All");
    return 0;
}
```

Listing 3: Reading file in ROOT format with HepMC3

MC events for ZEUS simulation

One of options for MC simulations in ZEUS was to use ASCII input very similar to HEPEVT format. The implementation of format in HepMC3 is listed below.

```
#include "HepMC3/WriterHEPEVT.h"
namespace HepMC3 {
class WriterHEPEVTZEUS : public  WriterHEPEVT {
public:
    WriterHEPEVTZEUS(const std::string &filename);
    void write_hepevt_event_header();
    void write_hepevt_particle( int index, bool iflong );
};
}
```

Listing 4: Header

```
#include "WriterHEPEVTZEUS.h"
#include "HepMC3/HEPEVT_Wrapper.h"
namespace HepMC3{
WriterHEPEVTZEUS::WriterHEPEVTZEUS(const std::string &filename):WriterHEPEVT(filename) {}
void WriterHEPEVTZEUS::write_hepevt_event_header()
{    fprintf(m_file," E % 12i% 12i% 12i\n",HEPEVT_Wrapper::event_number(),0,HEPEVT_Wrapper::number_entries());}
  void WriterHEPEVTZEUS::write_hepevt_particle( int index, bool /*iflong*/ ){
    fprintf(m_file,"% 12i% 8i",HEPEVT_Wrapper::status(index), HEPEVT_Wrapper::id(index));
    fprintf(m_file,"% 8i% 8i",HEPEVT_Wrapper::first_parent(index),HEPEVT_Wrapper::last_parent(index));
    fprintf(m_file,"% 8i% 8i",HEPEVT_Wrapper::first_child(index),HEPEVT_Wrapper::last_child(index));
    fprintf(m_file,      "% 19.11E% 19.11E% 19.11E% 19.11E% 19.11E\n",HEPEVT_Wrapper::px(index),HEPEVT_Wrapper::py(index),
        HEPEVT_Wrapper::pz(index),HEPEVT_Wrapper::e(index),HEPEVT_Wrapper::m(index));
    fprintf(m_file, "%-52s% 19.11E% 19.11E% 19.11E% 19.11E% 19.11E\n"," ",HEPEVT_Wrapper::x(index),HEPEVT_Wrapper::y(index),
        HEPEVT_Wrapper::z(index),HEPEVT_Wrapper::t(index),0.0);
}
}// namespace HepMC3
```

Listing 5: Source

Installation on CentOS7

```
[root@hostname1 username]# yum install HepMC3 HepMC3-devel  HepMC3-search HepMC3-search-devel  HepMC3-rootIO HepMC3-rootIO-devel HepMC3-interfaces-devel HepMC3-doc
Resolving Dependencies
Skipping filters plugin, no data
--> Running transaction check
---> Package HepMC3.x86_64 0:3.1.0-3.el7 will be installed
---> Package HepMC3-devel.x86_64 0:3.1.0-3.el7 will be installed
---> Package HepMC3-doc.noarch 0:3.1.0-3.el7 will be installed
---> Package HepMC3-interfaces-devel.noarch 0:3.1.0-3.el7 will be installed
---> Package HepMC3-rootIO.x86_64 0:3.1.0-3.el7 will be installed
---> Package HepMC3-rootIO-devel.x86_64 0:3.1.0-3.el7 will be installed
---> Package HepMC3-search.x86_64 0:3.1.0-3.el7 will be installed
---> Package HepMC3-search-devel.x86_64 0:3.1.0-3.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

==========================================================================================================================================================================
 Package                                                    Arch                                        Version                                          Repository                                   S
==========================================================================================================================================================================
Installing:
 HepMC3                                                     x86_64                                      3.1.0-3.el7                                      epel-testing                                 21
 HepMC3-devel                                               x86_64                                      3.1.0-3.el7                                      epel-testing                                  5
 HepMC3-doc                                                 noarch                                      3.1.0-3.el7                                      epel-testing                                  4
 HepMC3-interfaces-devel                                    noarch                                      3.1.0-3.el7                                      epel-testing                                  2
 HepMC3-rootIO                                              x86_64                                      3.1.0-3.el7                                      epel-testing                                  4
 HepMC3-rootIO-devel                                        x86_64                                      3.1.0-3.el7                                      epel-testing                                 5.
 HepMC3-search                                              x86_64                                      3.1.0-3.el7                                      epel-testing                                  2
 HepMC3-search-devel                                        x86_64                                      3.1.0-3.el7                                      epel-testing                                  1

Transaction Summary
==========================================================================================================================================================================
Install  8 Packages

Total download size: 44 M
Installed size: 85 M
Is this ok [y/d/N]: y
Downloading packages:
HepMC3-doc-3.1.0-3.el7.noarch.rpm                                                                                                                                  |  43 MB  00:00:01
(1/7): HepMC3-3.1.0-3.el7.x86_64.rpm                                                                                                                               | 218 kB  00:00:00
(2/7): HepMC3-devel-3.1.0-3.el7.x86_64.rpm                                                                                                                         |  52 kB  00:00:00
(3/7): HepMC3-interfaces-devel-3.1.0-3.el7.noarch.rpm                                                                                                              |  21 kB  00:00:00
(4/7): HepMC3-rootIO-3.1.0-3.el7.x86_64.rpm                                                                                                                        |  43 kB  00:00:00
(5/7): HepMC3-rootIO-devel-3.1.0-3.el7.x86_64.rpm                                                                                                                  | 5.5 kB  00:00:00
(6/7): HepMC3-search-3.1.0-3.el7.x86_64.rpm                                                                                                                        |  29 kB  00:00:00
(7/7): HepMC3-search-devel-3.1.0-3.el7.x86_64.rpm                                                                                                                  |  10 kB  00:00:00
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------
Total                                                                                                                                                  874 kB/s | 378 kB  00:00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : HepMC3-3.1.0-3.el7.x86_64
  Installing : HepMC3-devel-3.1.0-3.el7.x86_64
  Installing : HepMC3-search-3.1.0-3.el7.x86_64
  Installing : HepMC3-rootIO-3.1.0-3.el7.x86_64
  Installing : HepMC3-rootIO-devel-3.1.0-3.el7.x86_64
  Installing : HepMC3-search-devel-3.1.0-3.el7.x86_64
  Installing : HepMC3-interfaces-devel-3.1.0-3.el7.noarch
  Installing : HepMC3-doc-3.1.0-3.el7.noarch
  Verifying  : HepMC3-devel-3.1.0-3.el7.x86_64
  Verifying  : HepMC3-3.1.0-3.el7.x86_64
  Verifying  : HepMC3-search-devel-3.1.0-3.el7.x86_64
  Verifying  : HepMC3-search-3.1.0-3.el7.x86_64
  Verifying  : HepMC3-doc-3.1.0-3.el7.noarch
  Verifying  : HepMC3-rootIO-devel-3.1.0-3.el7.x86_64
  Verifying  : HepMC3-rootIO-3.1.0-3.el7.x86_64
  Verifying  : HepMC3-interfaces-devel-3.1.0-3.el7.noarch

Installed:
  HepMC3.x86_64 0:3.1.0-3.el7         HepMC3-devel.x86_64 0:3.1.0-3.el7         HepMC3-doc.noarch 0:3.1.0-3.el7  HepMC3-interfaces-devel.noarch 0:3.1.0-3.el7  HepMC3-rootIO.x86_64 0:3.1.0-3.el7  HepMC3-rootIO-devel.x86_64 0:3.1.0-3.e
  HepMC3-search.x86_64 0:3.1.0-3.el7  HepMC3-search-devel.x86_64 0:3.1.0-3.el7

Complete!
New leaves:
  HepMC3-doc.noarch
  HepMC3-interfaces-devel.noarch
  HepMC3-rootIO-devel.x86_64
  HepMC3-search-devel.x86_64
```

Installation of MC stack, e.g. including Rivet

As any problem, the installation of software has standard solutions of "enterprise" level. Rivet and many MC tools can be installed within minutes.

- Install Fedora/CentOS/RHEL/openSUSE
- Do as root in, e.g. Fedora

```
[root@fedora ~]$ dnf -y install dnf-plugins-core
[root@fedora ~]$ dnf -y copr enable averbyts/HEPrpms
[root@fedora ~]$ dnf -y install Rivet
```

- Docs at https://github.com/andriish/HEPrpms

Adoption of mainstream Linux tools solves the complexity almost immediately.

I/O: ASCII formats

```
HepMC::Version 3.0.0
HepMC::Asciiv3-START_EVENT_LISTING
W 0
E 0 7 12
U GEV MM
W 1.00000000000000000000e+00
A 0 GenCrossSection 2.64422551e+03 2.64422551e+03 -1 -1
A 0 GenPdfInfo 11 -11 9.97420767e-01 9.99999975e-01 9.18812775e+01 1.56824725e+01 2.82148362e+06 0 0
A 0 alphaQCD 0.129844
A 0 alphaQED 0.007818181
A 0 event_scale 91.88128
A 0 mpi 0
A 0 signal_process_id 221
A 0 signal_process_vertex 0
P 1 0 11 0.0000000000000000e+00 0.0000000000000000e+00 4.5999999997161737e+01 4.6000000000000007e+01 5.1099999999999995e-04 4
P 2 1 11 0.0000000000000000e+00 0.0000000000000000e+00 4.5881355265109356e+01 4.5881355265109356e+01 0.0000000000000000e+00 61
P 3 1 22 0.0000000000000000e+00 0.0000000000000000e+00 1.1864473489064407e-01 1.1864473489064407e-01 0.0000000000000000e+00 1
P 4 0 -11 0.0000000000000000e+00 0.0000000000000000e+00 -4.5999999997161737e+01 4.6000000000000007e+01 5.1099999999999995e-04 4
P 5 4 -11 0.0000000000000000e+00 0.0000000000000000e+00 -4.5999998855671230e+01 4.5999998855671230e+01 0.0000000000000000e+00 61
P 6 4 22 0.0000000000000000e+00 0.0000000000000000e+00 -1.1443287704082650e-06 1.1443287704082650e-06 0.0000000000000000e+00 1
P 7 2 11 0.0000000000000000e+00 0.0000000000000000e+00 4.5881355265109356e+01 4.5881355265109356e+01 0.0000000000000000e+00 21
```

Listing 7: HepMC3 Asciiv3 (HepMC3 standard)

```
HepMC::Version 3.0.0
HepMC::IO_GenEvent-START_EVENT_LISTING
E 0 0 9.188128e+01 1.298440e-01 7.818181e-03 221 0 7 10001 10004 0 1 1.0000000000000000e+00
N 1 "0"
U GEV MM
C 2.6442255100000002e+03 2.6442255100000002e+03
F 11 -11 9.97420767e-01 9.99999975e-01 9.18812775e+01 1.56824725e+01 2.82148362e+06 0 0
V -1 0 0 0 0 0 1 2 0
P 10001 11 0.0000000000000000e+00 0.0000000000000000e+00 4.5999999997161737e+01 4.6000000000000007e+01 5.1099999999999995e-04 4 0 0 -1 0
P 10002 11 0.0000000000000000e+00 0.0000000000000000e+00 4.5881355265109356e+01 4.5881355265109356e+01 0.0000000000000000e+00 61 0 0 -3 0
P 10003 22 0.0000000000000000e+00 0.0000000000000000e+00 1.1864473489064407e-01 1.1864473489064407e-01 0.0000000000000000e+00 1 0 0 0 0
V -2 0 0 0 0 0 1 2 0
P 10004 -11 0.0000000000000000e+00 0.0000000000000000e+00 -4.5999999997161737e+01 4.6000000000000007e+01 5.1099999999999995e-04 4 0 0 -2 0
P 10005 -11 0.0000000000000000e+00 0.0000000000000000e+00 -4.5999998855671230e+01 4.5999998855671230e+01 0.0000000000000000e+00 61 0 0 -4 0
P 10006 22 0.0000000000000000e+00 0.0000000000000000e+00 -1.1443287704082650e-06 1.1443287704082650e-06 0.0000000000000000e+00 1 0 0 0 0
V -3 0 0 0 0 0 0 1 0
```

Listing 8: IO_GenEvent (HepMC2 standard) as implemented in HepMC3

```
E 0 12
 4 -11 0 0 3 11 0.00000000E+00 0.00000000E+00 -4.60000000E+01 4.60000000E+01 5.11000000E-04
       0.00000000E+00 0.00000000E+00 0.00000000E+00 0.00000000E+00
 4 11 0 0 4 12 0.00000000E+00 0.00000000E+00 4.60000000E+01 4.60000000E+01 5.11000000E-04
       0.00000000E+00 0.00000000E+00 0.00000000E+00 0.00000000E+00
61 -11 1 1 5 5 0.00000000E+00 0.00000000E+00 -4.59999989E+01 4.59999989E+01 0.00000000E+00
       0.00000000E+00 0.00000000E+00 0.00000000E+00 0.00000000E+00
61 11 2 2 6 6 0.00000000E+00 0.00000000E+00 4.58813553E+01 4.58813553E+01 0.00000000E+00
       0.00000000E+00 0.00000000E+00 0.00000000E+00 0.00000000E+00
21 -11 3 3 7 7 0.00000000E+00 0.00000000E+00 -4.59999989E+01 4.59999989E+01 0.00000000E+00
```

We keep compatibility! Note

Listing 9: HepEVT (Fortran era generators) as implemented in HepMC3

absence of unused color flow information in Asciiv3.

I/O: LHEF

- The original Les Houches accord event format (LHEF) is designed for communicating between Matrix element generators and MCEG.
- It agreed upon in 2001 [18](see updates in Refs. [19][20][21]).
- **The routines to handle LHEF is a precious part of HepMC3.**

```
<LesHouchesEvents version="3.0">
  <header>
  <MG5ProcCard>
#*************************************************************
#*                     MadGraph 5                            *
#*                                                           *
#*               *                       *                   *
#*                 *        * *        *                     *
#*                   * * * * 5 * * * *                       *
#*                 *        * *        *                     *
#*               *                       *                   *
#*                                                           *
#*                                                           *
#*         VERSION 2.0.0.beta4           2013-06-22          *
#*                                                           *
#*    The MadGraph Development Team - Please visit us at     *
#*    https://server06.fynu.ucl.ac.be/projects/madgraph      *
#*                                                           *
#*************************************************************
#*                                                           *
```

Listing 10: First lines of LHEF header

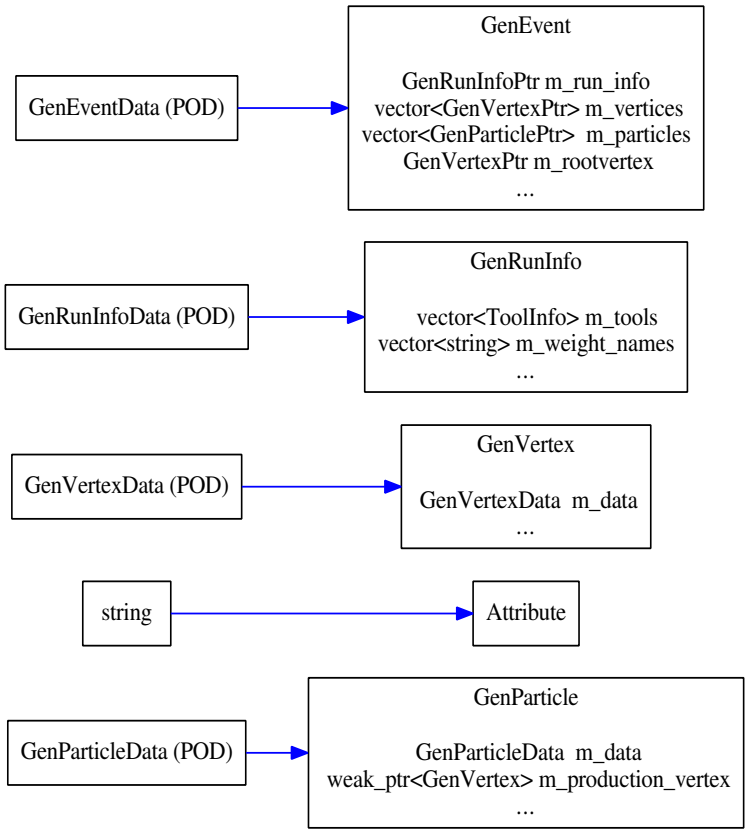
LHEF is basically XML with extra rules.

```
  <event>
6     66 0.50109093E+02 0.88344569E+02 0.75563862E-02 0.12114027E+00
      2 -1    0    0  502    0 0.00000000E+00 0.00000000E+00 0.62728157E+03 0.62728166E+03 0.33000000E+00 0.0000E+00 0.0000E+00
      5 -1    0    0  501    0 0.00000000E+00 0.00000000E+00 -.44481443E+02 0.44739678E+02 0.48000000E+01 0.0000E+00 0.0000E+00
      6  2    1    2  501    0 -.59350665E+02 0.72244533E+02 0.21037840E+03 0.28804239E+03 0.17311146E+03 0.0000E+00 0.0000E+00
     24  1    3    3    0    0 0.28747403E+02 0.66692858E+02 0.11544955E+03 0.15832494E+03 0.80398000E+02 0.0000E+00 0.0000E+00
      5  1    3    3  501    0 -.88098069E+02 0.55516749E+01 0.94928843E+02 0.12971745E+03 0.48000000E+01 0.0000E+00 0.0000E+00
      1  1    1    2  502    0 0.59350665E+02 -.72244533E+02 0.37242173E+03 0.38397894E+03 0.33000000E+00 0.0000E+00 0.0000E+00
#aMCatNLO 1  6  2  0  0 0.00000000E+00 0.00000000E+00 9  0  0 0.10000000E+01 0.86321681E+00 0.11022720E+01 0.00000000E+00 0.00000000E+00
#aMCatNLO 1  6  2  0  0 0.00000000E+00 0.00000000E+00 9  0  0 0.10000000E+01 0.91292678E+00 0.10493312E+01 0.00000000E+00 0.00000000E+00
#aMCatNLO 1  6  2  0  0 0.00000000E+00 0.00000000E+00 9  0  0 0.10000000E+01 0.91292678E+00 0.10493312E+01 0.00000000E+00 0.00000000E+00
#aMCatNLO 1  6  2  0  0 0.00000000E+00 0.00000000E+00 9  0  0 0.10000000E+01 0.91292678E+00 0.10493312E+01 0.00000000E+00 0.00000000E+00
  <rwgt>
   <wgt id="1001"> 0.50109E+02 </wgt>
   <wgt id="1002"> 0.43255E+02 </wgt>
   <wgt id="1003"> 0.55234E+02 </wgt>
```

Listing 11: First lines of some LHEF event

More:

I/O: ROOT Tree format



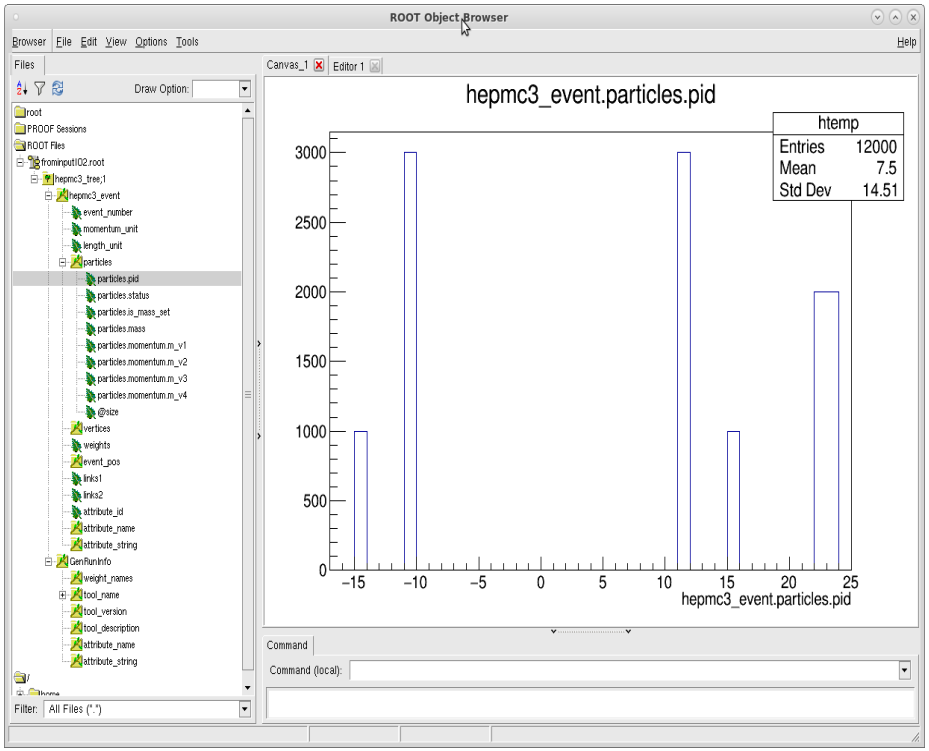


Figure: HepMC3 ROOT file in TBrowser.

HepMC3 uses custom streamers + POD types for ROOT I/O. **HepMC3 ROOT files are readable with standard ROOT, NO externals needed.**

I/O: code example

```
#include "HepMC3/GenEvent.h"
#include "HepMC3/WriterAscii.h"
#include "HepMC3/ReaderAsciiHepMC2.h"
int main()
{
    HepMC3::ReaderAsciiHepMC2 inputA("inputIO1.hepmc");
    if(inputA.failed()) return 1;
    HepMC3::WriterAscii       outputA("frominputIO1.hepmc");
    if(outputA.failed()) return 2;
    while( !inputA.failed() )
        {
            HepMC3::GenEvent evt(HepMC3::Units::GEV,HepMC3::Units::MM);
            inputA.read_event(evt);
            if( inputA.failed() )  {printf("End of file reached. Exit.\n"); break;}
            outputA.write_event(evt);
            evt.clear();
        }
    inputA.close();
    outputA.close();
    return 0;
}
```

Listing 12: Reading file in HepMC2 format (IO_GenEvent) and writing in HepMC3 (Asciiv3)

- **Note: Works in the same way for all formats in different combinations.**

The examples in HepMC3 provide a simple event browser

Figure: Pythia 8.2.40 simulated process $e^+e^- \to \tau^+\tau^-$ with ISR and w/o hadronisation and decays. The display shows event information (e.g. run and event number, ...), depicts vertices as ellipses and the particles as arrows. The information on vertices and particles (ids and flavor) are given as well. The right panel shows selected distributions of physical quantities in the event, e.g. transverse momentum.

Search engine

The main class to perform selection is HepMC3::Selector.

```
ConstSelectorPtr status = std::make_shared<SelectorWrapper<int> >([](ConstParticlePtr p)->int{return p->status();});
ConstSelectorPtr pt = std::make_shared<SelectorWrapper<double> >([](ConstParticlePtr p)->double{return p->momentum().pt();});
```

One can then use the Selector to construct Filter functions that evaluate on particles, e.g.

```
Filter is_stable = (*status) == 1;
bool stable = is_stable(p);
bool beam = (*status == 4)(p);
```

Selector contains a few standard Selectors already defined, e.g.

```
ConstGenParticlePtr p;
(Selector::STATUS == 1)(p);
(Selector::PT > 15.)(p);
(abs(Selector::RAPIDITY) < 2.5)(p);
```

One can also combined them e.g.

```
Filter myCuts = (Selector::PT > 15.) && (*abs(Selector::RAPIDITY) < 2.5) || (Selector::PT > 100.);
bool passCuts = myCuts(p);
```

Advantages of C++11 in action.

Bibliography I

[1] A. Buckley, P. Ilten, D. Konstantinov, L. Lonnblad, J. Monk, W. Porkorski, T. Przedzinski and A. Verbytskyi,
The HepMC3 event record library for Monte Carlo event generators.
Comput. Phys. Commun. **260**, 107310 (2021).
arXiv:1912.08005.

[2] M. Dobbs and J.B. Hansen,
The HepMC C++ Monte Carlo event record for High Energy Physics.
Comput. Phys. Commun. **134**, 41 (2001).

[3] T. Sjöstrand, S. Mrenna and P.Z. Skands,
A brief introduction to PYTHIA 8.1.
Comput. Phys. Commun. **178**, 852 (2008).
arXiv:0710.3820.

[4] J. Bellm et al.,
Herwig 7.0/Herwig++ 3.0 release note.
Eur. Phys. J. **C76**, 196 (2016).
arXiv:1512.01178.

[5] T. Gleisberg et al.,
Event generation with SHERPA 1.1.
JHEP **02**, 007 (2009).
arXiv:0811.4622.

[6] GEANT4, S. Agostinelli et al.,
GEANT4: A Simulation toolkit.
Nucl. Instrum. Meth. **A506**, 250 (2003).

[7] A. Ferrari et al.,
FLUKA: A multi-particle transport code (Program version 2005).
(2005).

Bibliography II

[8] I. Antcheva et al.,
ROOT: A C++ framework for petabyte data storage, statistical analysis and visualization.
Comput. Phys. Commun. **182**, 1384 (2011).

[9] A. Buckley et al.,
Rivet user manual.
Comput. Phys. Commun. **184**, 2803 (2013).
arXiv:1003.0694.

[10] Wenzel Jakob and Jason Rhinelander and Dean Moldovan,
pybind11 – Seamless operability between C++11 and Python, 2017.
Https://github.com/pybind/pybind11.

[11] The binder development team,
binder.
Https://github.com/RosettaCommons/binder.

[12] The bxzstr development team,
bxzstr.
Https://github.com/tmaklin/bxzstr.

[13] The pyhepmc development team,
pyhepmc.
Https://github.com/scikit-hep/pyhepmc.

[14] Gardiner, S. and Isaacson, J. and Pickering, L.,
NuHepMC: A standardized event record format for neutrino event generators.
(2023).
arXiv:2310.13211.

Bibliography III

[15] J.G. Milhano et al.,
Lisbon Accord: a standard format for heavy-ion event generators.
(2017).

[16] The pandas development team,
pandas-dev/pandas: Pandas.
Https://doi.org/10.5281/zenodo.3509134.

[17] The uproot development team,
uproot.
Https://github.com/scikit-hep/uproot.

[18] E. Boos et al.,
Generic user process interface for event generators.
(2001).
arXiv:hep-ph/0109068.

[19] J. Alwall et al.,
A Standard format for Les Houches event files.
Comput. Phys. Commun. **176**, 300 (2007).
arXiv:hep-ph/0609017.

[20] J.M. Butterworth et al.,
THE TOOLS AND MONTE CARLO WORKING GROUP Summary Report from the Les Houches 2009 Workshop on TeV Colliders.
(2010).
arXiv:1003.1643.

[21] J.R. Andersen et al.,
Les Houches 2013: Physics at TeV Colliders: Standard Model Working Group Report.
(2014).
arXiv:1405.1067.