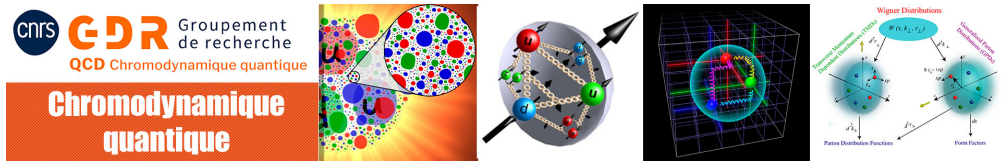


HEPMC Standards and the Path Forward

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

hepmc-dev@cern.ch

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

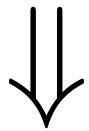
[a] Max-Planck-Institut für Physik (Werner-Heisenberg-Institut)

GDR-QCD seminar: introduction to Rivet, HepMC and selected expertises,

Saclay, July 01, 2021 (ZOOM)

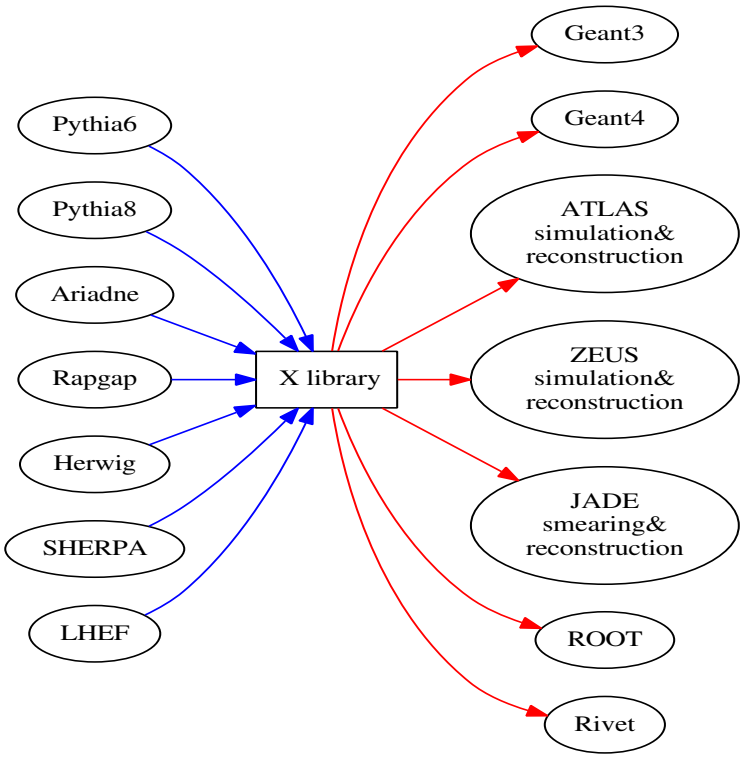
What is HepMC3: Definition and application

- HepMC3 is a library designed to operate with Monte Carlo event records in High Energy physics (HEP), see Ref. [1].
- Event record contains all physical information on the initial, intermediate and final state particles in the simulated collision: particle flavours, momenta, production coordinates, etc.



- 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[2], Herwig7 [3], SHERPA [4]), simulation software (e.g. Geant4 [5], FLUKA [6]), analysis frameworks (ROOT [7], experiment-specific), plotting tools (e.g. Rivet [8], HZTOOL [9]) etc.

What is HepMC3: Typical application example



- The typical task is to pass MC generated events between different generators, reconstruction programs and/or analysis frameworks.
- Before the HepMC3, the HepMC2 [10] partially performed these tasks.

Motivation to update HepMC2

- **The communication with users was interrupted during the changes in the developers team and platform migrations in CERN**.
- Maintenance:
  - HepMC2 is not actively developed for a long time, last minor release is in **2012**, many users have to apply patches.
  - Many small changes would break compatibility but will not allow significant design improvements.
- Features:
  - Physics-related issues should be resolved to meet requirements of **modern physics**, e.g. Heavy Ion information [11], event weight treatment, etc.
  - I/O capabilities should be modernised, e.g. ROOT, LHEF [12], serialisation of custom information.

HepMC3 is a natural successor of HepMC2. With an idea to keep what is good in HepMC2 and add more.

New in HepMC3 (vs. HepMC2)

- Consistent I/O for all popular formats of event records.
- The original LHEF routines by Leif Lönnblad.
- ROOT interface out of the box.
- Python2/Python3/PyPy interface out of the box.
- Extendable event record.
- Search engine to navigate in the event record.
- Multi-threading/thread safety.
- CMake instead of autotools.
- Automatic memory management with smart pointers.
- GitLab with CI and tickets.
- Test engine based on CTest.
- Extensive documentation and examples.
- Implemented interfaces in most modern MCEGs.

Current usage

User	Type of SW	HepMC3 interface implementation
Athena@ATLAS	Main SW of experiment	3.2.3 in a separate branch
SHERPA-MC	MCEG	3.1+ in SHERPA-MC 2.2.5+
ThePEG	MCEG	3.1 in ThePEG 2.2.0
Herwig7	MCEG	3.1 via ThePEG
Pythia8	MCEG	3.0+ in Pythia8.3 and in HepMC3
Pythia6	MCEG	3.1+ in HepMC3 examples
Tauola	MCEG	3.1+ in Taoula 3.64 and in HepMC3
Photos	MCEG	3.1+ in Photos 1.1.8 and in HepMC3
WHIZARD	MCEG	3.1+ in Whizard 2.8.2+
EvtGen	MCEG	3.1+ in EvtGen 2.0.0+
GeantV	Simulation	3.0
ACTS	Tracking	3.2+ in multiple versions
Rapgap	MCEG	3.1+, in a a merge request to master
Cascade	MCEG	3.1+, in a merge request to master
Rivet	Analysis/Plotting	3.2.0 in Rivet 3.1.0+
MC-TESTER	Analysis/Plotting	3.1+ in MC-TESTER 1.25.1 and in HepMC3

HepMC3 3.1+ and HepMC2 can co-exist in one installation.

Selected new features: Attributes in HepMC3

- An attribute is class that holds information on event, particle or vertex and can be represented as a string.
- Every attribute class should inherit from HepMC3::Attribute. The attribute is handled all the time as a string (in memory and in I/O operations) till an explicit request to convert it to an object of some type.

Example of attributes are given below

```
GenCrossSection 2.64422551e+03 2.64422551e+03 -1 -1
```

```
GenPdfInfo 11 -11 9.97420767e-01 9.99999975e-01 9.18812775e+01 1.56824725e+01 2.82148362e+06 0 0
```

```
alphaQCD 0.129844
```

These attributes represent information of classes

```
HepMC3::GenCrossSection
HepMC3::GenPdfInfo
double
```

One can add arbitrary complex information to event record.

Selected new features: separation of disk-resident formats and HepMC3::GenEvent

- The 'HepMC3::GenEvent' objects in memory are constructed from the information on disk, but **"Events on disk"** $\neq$ **"Events in memory"**.
- The same input file processed with different 'HepMC3::Reader' can produce different events. If e.g. only the stable particles in the final state are needed – no need to invent new formats. A custom reader will do the job on the standard files.

Selected new features: custom readers/writers for different formats

HepMC3 I/O is easily extendable for different purposes

- Simple example to write into dot format and use with GraphViz.

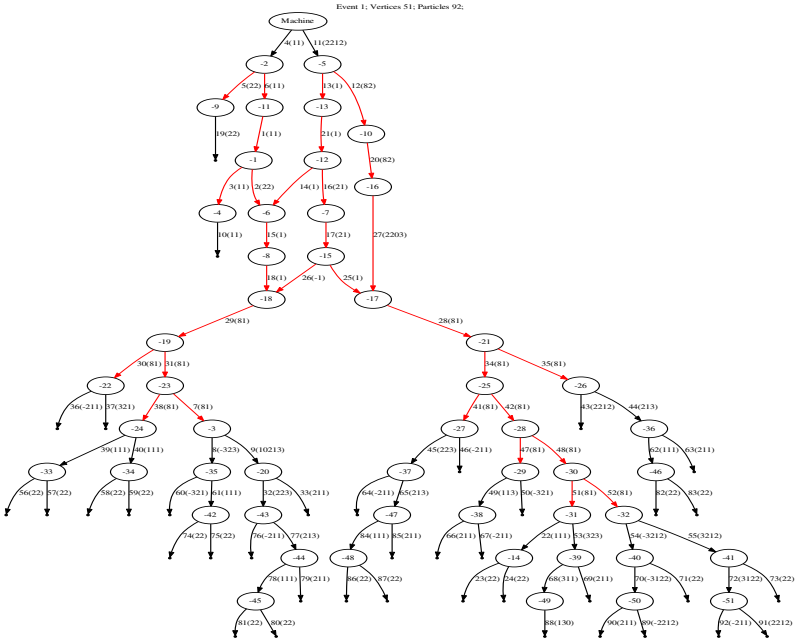


Figure: $e^- p$ collision in Herwig 7.1.4

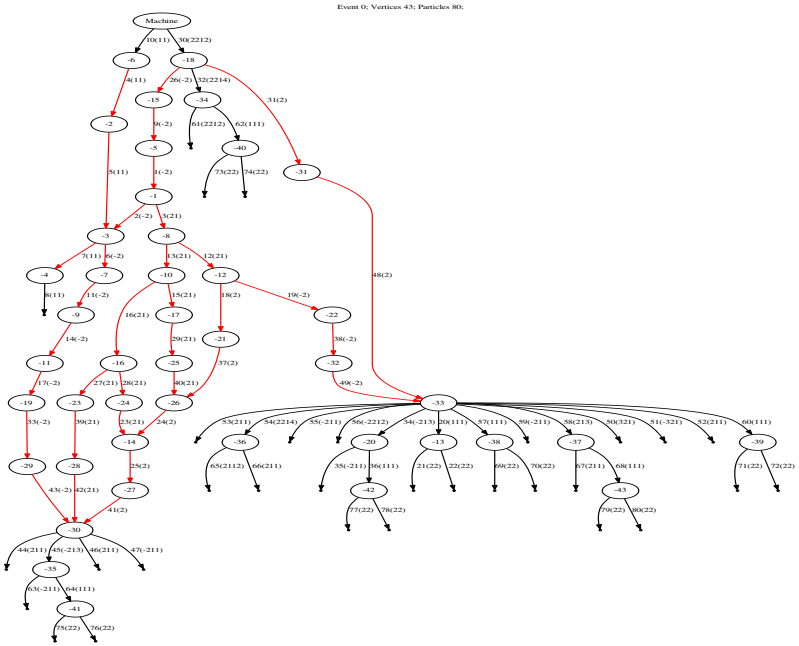


Figure: $e^- p$ collision in Pythia 8.2.40

Usage: the “classical” MCEGs and the simulations for old experiments

Because it is very easy to manipulate with different file formats in HepMC3, it is very easy to

- Interface a classical 20+ y.o. MCEG to modern tools.
- Interface a modern MCEG to 20+ y.o. detector simulation.

Usage: personal experience for QCD studies

The typical workflow for personal QCD studies:

- Run MCEG with HepMC3/ROOT support and produce HepMC3/ROOTTree events. $\approx 10^{7-8}$ events for $e^+e^-/e^\pm p$ or 10^{6-7} for pp. The ROOTTree output is compact and is stored locally or e.g. in dCache.
- Run ROOT analysis on the files (HepMC3 is NOT needed at this point) using PROOF. The input can be local or remote. With a 64 core machine, the typical QCD analysis runs at 10^5 events per second. I/O speed is not a problem. All the samples can be processed in $\mathcal{O}(1)$ hours.
- Use the filled histograms.

Usage: HepMC3 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.

Usage: Rivet with HepMC3

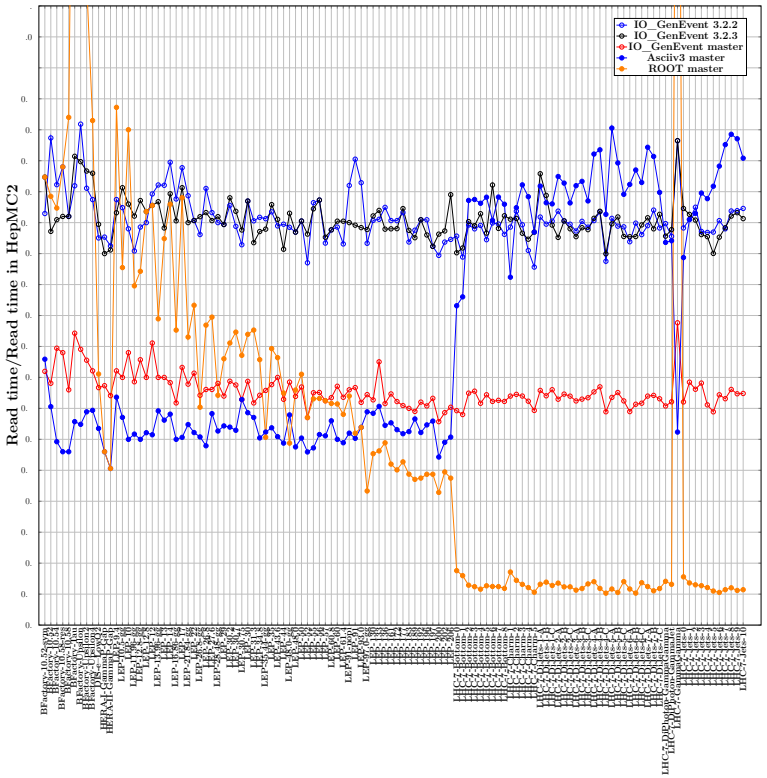
If compiled with HepMC3>3.2.0, Rivet uses standard HepMC3 functions

```
1  std::shared_ptr<HepMC3::Reader> deduce_reader(const std::string &filename);
   std::shared_ptr<HepMC3::Reader> deduce_reader(std::istream &stream);
```

to deal with input, i.e. it is **disk format-agnostic** and the format of input will be deduced automatically. So, if HepMC3 can read some format, Rivet will be able to do it as well. No need for separate "converters" that use disk and Rivet can read ROOT.

Developments

HepMC3 is stable and ready for use for a long time. But it does not mean the development is limited now to documentation improvements and bugfixes. **User issues should be adressed!** →



Improvements of I/O speed in master as a result of discussions with users.

Bonus slide: HepMC3+DIS+Rivet, part I

The first step to use Rivet with any MCEG is to assure the MCEG can produce an output readable by Rivet. **Means the MCEG should have HepMC interface**.
Right now, codes under review by the MCEG authors:

- HepMC3 interface for CASCADE
- HepMC3 interface for RAPGAP

Once HepMC interface is accepted in the codebase, the implementation of Rivet interface is trivial.

Bonus slide: HepMC3+DIS+Rivet, part II

So, everything is great for DIS@Rivet?
Not really...

- DIS has specific issues related to the definitions of kinematic variables, which are not always easy to propagate to MC event record.
- The interface to actually used DIS MCEGs are missing.
- Even very good modern MCEGs have issues with certain aspects of ep physics, e.g. beam remnants.
- Expertise is scarce while the Rivet codes for HERA should be submitted voluntarily (v.s. obligatory for LHC).

But, actually the invitation to this workshop stimulated a closer collaboration with Rivet authors. Thank you! This should work :)

Dialogue with GDR-QCD

- 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!** .
- If your idea was implemented, the issue was resolved, a bug was fixed – please give some **feedback**, close the resolved issues etc.

Summary

- HepMC3 3.2.4 will be released very soon. Another step towards full replacement of HepMC2, e.g. in ATLAS and MCEG community.
- Collecting feedback from LHC experiments, MCEG authors and interested individuals. **Discussions during this workshop will provide a crucial information.**

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

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

Backup slides

Obsolete and removed features (vs. HepMC2)

- General features and I/O:
  - Some duplicated functions.
  - Operators '«' and '»' for event input/output that violated class hierarchy.
  - Custom iterator-based access. Finally usage of STL is possible!
  - Flow and Polarisation classes are removed and the information should be represented by generic attributes. Event record is simplified!

Particle in standard HepMC2 representation (IO_GenEvent)

```
P 10001 11 0.0e+00 0.0e+00 4.59999e+01 4.60000e+01 5.10999e-04 4 0 0 -1 0
```

Particle in standard HepMC3 representation (Asciiv3)

```
1  P 1 0 11 0.0e+00 0.0e+00 4.59999e+01 4.60000e+01 5.10999e-04 4
```

The difference stands for polarisation and event flow information
$\rightarrow$ rarely filled and meaningful.

```
                                                    0 0 -1 0
```

Requirements for HepMC3 3.2.3+

Minimal:

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

Recommended=Minimal+

- ROOT6

Full=Recommended+

- doxygen, latex, graphviz for documentation
- valgrind, Pythia8, HepMC2, Photos [13], Tauola [14], MC-TESTER [15] for tests
- Fortran77 compiler for examples
- Python2/Python3/PyPy modules

Contact developers in case something is not working.

HepMC3 installation

There is a peception that the MCEG-related software stack is too complex and hard to install. Not the case for HepMC3:

OS	Repository	HepMC3 versions	Recommended for installation?	Credits
Fedora	Standard	last	Yes	Mattias Ellert
CentOS	EPEL	last	Yes	Mattias Ellert
MacOS	homebrew-hep	last	Yes	Enrico Bothmann/AV
ArchLinux	AUR	last	Yes	Frank Siegert
openSUSE	science	last	Yes	Atri Bhattacharya
Windows10	PyPi	last	Yes	AV
Debian, Ubuntu	Standard	3.1.1	Outdated, install from source	Mo Zhou
Others		last	Install from source	

One should be able to do

```
[username@fedora ~]$ dnf install HepMC3
or
[username@centos ~]$ yum install HepMC3
or
username—macbook%  brew install  HepMC3
```

without knowing the requirements

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 10: Reading file in ROOT format with HepMC3

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 11: 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 12: 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 13: 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 [16](see updates in Refs. [12][17][18]).
- **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 14: First lines of LHEF header

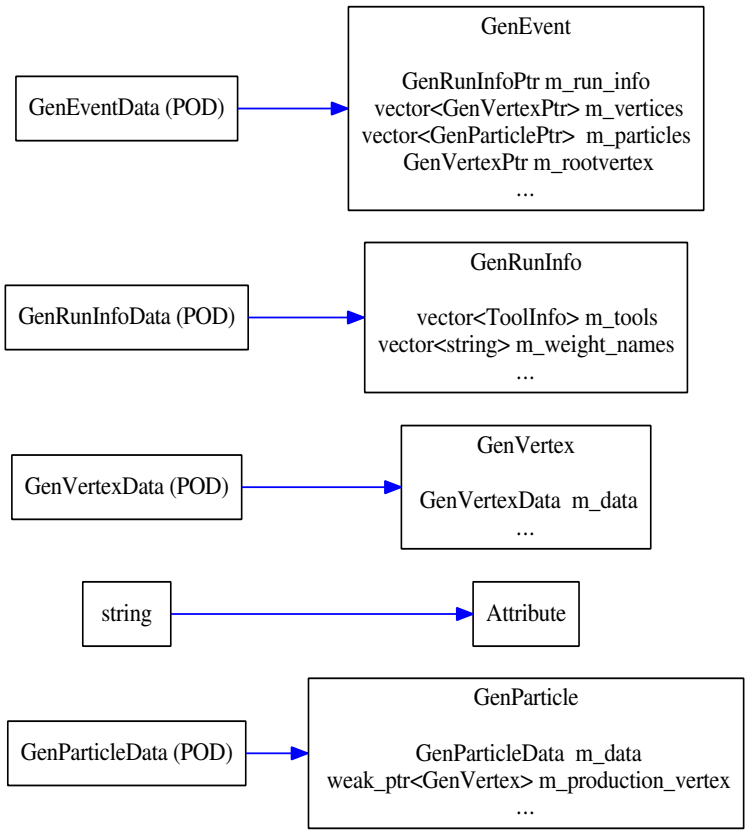
LHEF is basically XML with extra rules.

```
  <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>
```

More:

Listing 15: First lines of some LHE event

I/O: ROOT Tree format



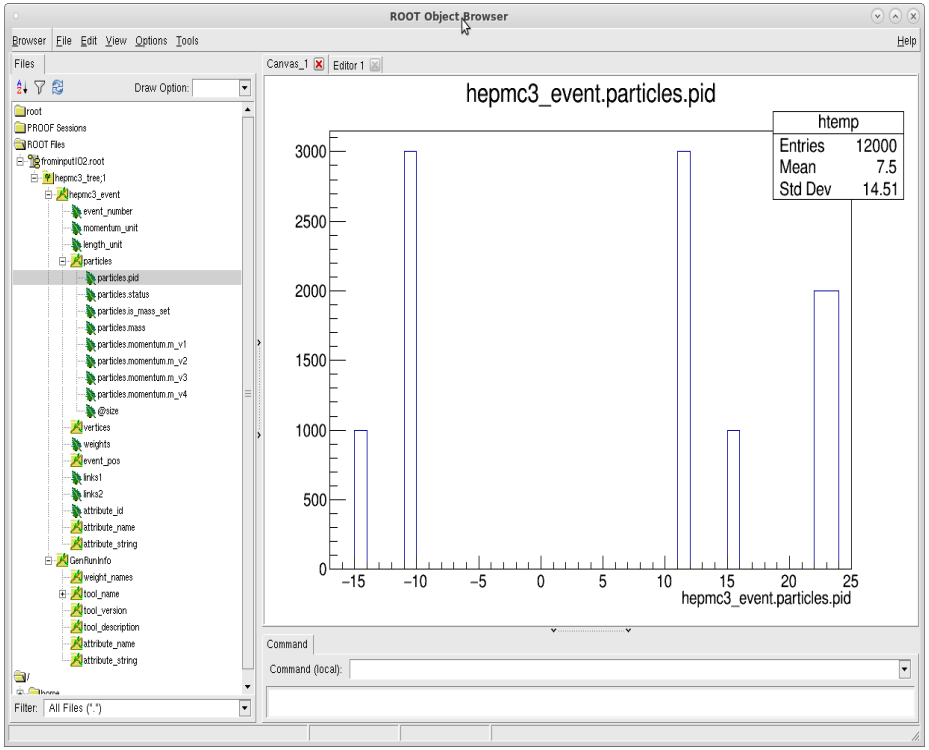


Figure: HepMC3 ROOT file in TBrowser.

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

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

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

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

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

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

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

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

Bibliography II

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

[9] J. Bromley et al.,
HZTOOL: A package for Monte Carlo-data comparison at HERA (version 1.0),
Future physics at HERA. Proceedings, Workshop, Hamburg, Germany, September 25, 1995-May 31, 1996.
Vol. 1, 2.
(1995).

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

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

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

[13] E. Barberio and Z. Was,
PHOTOS: A Universal Monte Carlo for QED radiative corrections. Version 2.0.
Comput. Phys. Commun. **79**, 291 (1994).

[14] S. Jadach, J.H. Kühn and Z. Was,
TAUOLA: a library of Monte Carlo programs to simulate decays of polarized tau leptons.
Comput. Phys. Commun. **64**, 275 (1990).

Bibliography III

[15] P. Golonka,T. Pierzchala and Z. Was,
MC-TESTER: A Universal tool for comparisons of Monte Carlo predictions for particle decays in high-energy physics.
Comput. Phys. Commun. **157**, 39 (2004).
arXiv:hep-ph/0210252.

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

[17] 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.

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