Pythia and how to use it

Olena Mezhenska

Particle physics workshop: from STAR to EIC

Feb 15 – 18, 2024

Outlook

- Introduction
  - What is PYTHIA?
  - Physical processes in PYHTIA
- Installation
- Program structure
- Settings
  - Beams
  - Process selection
- Analysis of generated event
  - The Vec4 class
  - The Particle class
  - The Event class
- Program output
- Example

What is PYTHIA?

- The Pythia program is a standard tool for the generation of events in high-energy collisions between elementary particles;

- Currently the beam particles must be either

  hadron pair, lepton pair, a lepton and a hadron;

  photon pair, or a photon and a hadron;

  heavy Ion beam.

Settings

The internal PYTHIA 8.3 event generation is divided into three steps:

- **Process level;**
- **Parton level;**
- **Hadron level;**

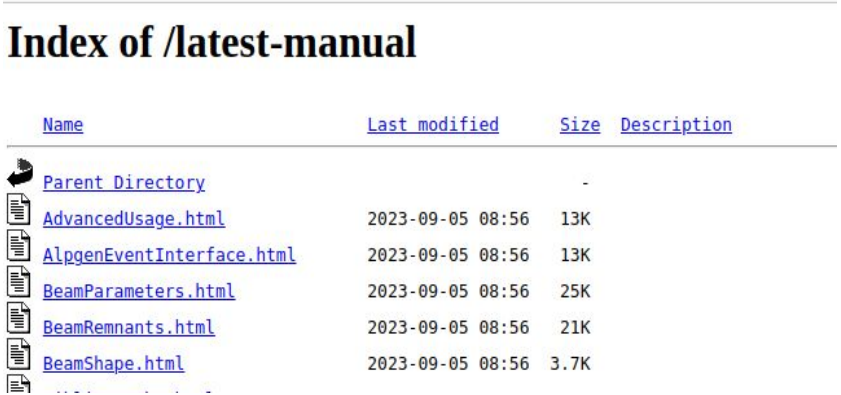
Settings

The internal PYTHIA 8.3 event generation is divided into three steps:

- **Process level;**
- **Parton level;**
- **Hadron level;**

All possible setting keys:

https://www.pythia.org/latest-manual/



Index of /latest-manual

Name	Last modified	Size	Description
Parent Directory		-	
AdvancedUsage.html	2023-09-05 08:56	13K	
AlpgenEventInterface.html	2023-09-05 08:56	13K	
BeamParameters.html	2023-09-05 08:56	25K	
BeamRemnants.html	2023-09-05 08:56	21K	
BeamShape.html	2023-09-05 08:56	3.7K	

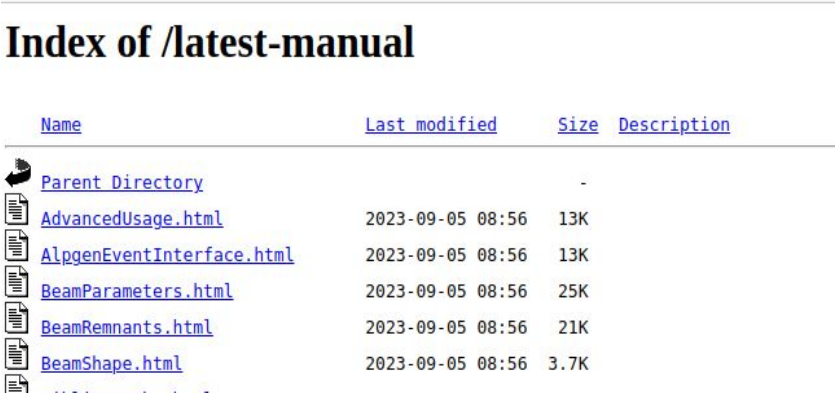
Settings

The internal PYTHIA 8.3 event generation is divided into three steps:

- **Process level;**
- **Parton level;**
- **Hadron level;**

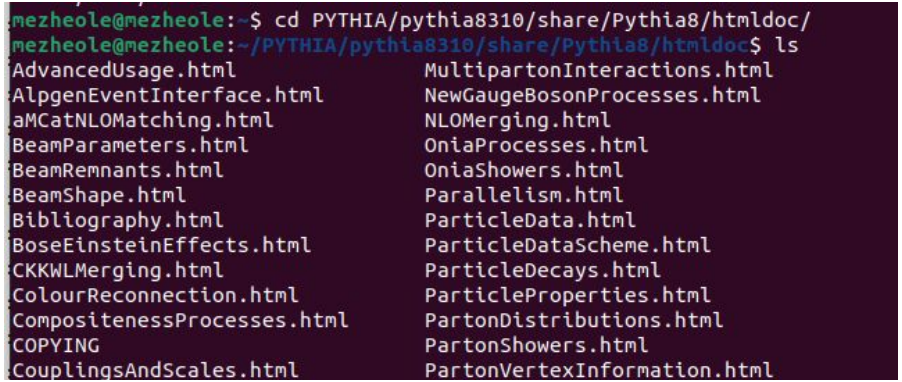
All possible setting keys:

https://www.pythia.org/latest-manual/ or ***cd share/Pythia8/htmldoc/Welcome.html***



Index of /latest-manual

Name	Last modified	Size	Description
Parent Directory		-	
AdvancedUsage.html	2023-09-05 08:56	13K	
AlpgenEventInterface.html	2023-09-05 08:56	13K	
BeamParameters.html	2023-09-05 08:56	25K	
BeamRemnants.html	2023-09-05 08:56	21K	
BeamShape.html	2023-09-05 08:56	3.7K	



```
mezheole@mezheole:~$ cd PYTHIA/pythia8310/share/Pythia8/htmldoc/
mezheole@mezheole:~/PYTHIA/pythia8310/share/Pythia8/htmldoc$ ls
AdvancedUsage.html                MultipartonInteractions.html
AlpgenEventInterface.html         NewGaugeBosonProcesses.html
aMCatNLOMatching.html             NLOMerging.html
BeamParameters.html               OniaProcesses.html
BeamRemnants.html                 OniaShowers.html
BeamShape.html                    Parallelism.html
Bibliography.html                 ParticleData.html
BoseEinsteinEffects.html          ParticleDataScheme.html
CKKWLMerging.html                 ParticleDecays.html
ColourReconnection.html           ParticleProperties.html
CompositenessProcesses.html       PartonDistributions.html
COPYING                           PartonShowers.html
CouplingsAndScales.html           PartonVertexInformation.html
```

Physics processes in PYTHIA

1. **QCD processes**

Physics processes in PYTHIA

1.

QCD processes

Soft QCD sub-processes

Hard QCD sub-processes

Physics processes in PYTHIA

1.

QCD processes

Soft QCD sub-processes

- Elastic ($AB \rightarrow AB$)
- Single diffractive

  ($AB \rightarrow XB$ *or* $AB \rightarrow AX$)
- Double diffractive ($AB \rightarrow XY$)
- Non-diffractive

Hard QCD sub-processes

Physics processes in PYTHIA

1.

QCD processes

Soft QCD sub-processes

- Elastic ($AB \rightarrow AB$)
- Single diffractive

  ($AB \rightarrow XB$ *or* $AB \rightarrow AX$)
- Double diffractive ($AB \rightarrow XY$)
- Non-diffractive

Hard QCD sub-processes

- Light Quarks and Gluons

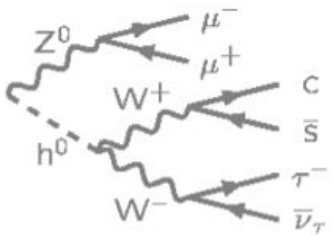
  $(gg \rightarrow gg; qg \rightarrow qg; qq \rightarrow qq; q\bar{q} \rightarrow gg; gg \rightarrow q\bar{q})$
- Heavy Flavours

  $(gg \rightarrow c\bar{c}; gg \rightarrow b\bar{b}; q\bar{q} \rightarrow c\bar{c})$
- Three-parton processes

  $(gg \rightarrow ggg; q\bar{q} \rightarrow ggg; \bar{q}\bar{q} \rightarrow \bar{q}\bar{q}g; gg \rightarrow q\bar{q}g)$

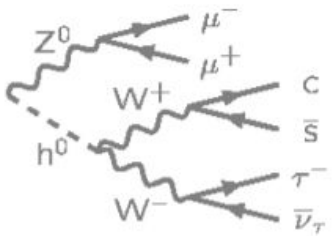
Physics processes in PYTHIA

2. Resonance decays

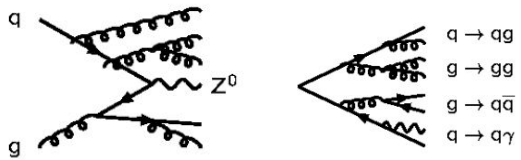


Physics processes in PYTHIA

2. Resonance decays

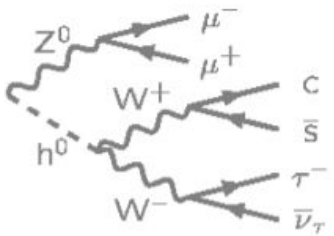


3. Parton showers (ISR, FSR)

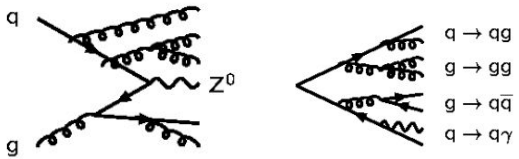


Physics processes in PYTHIA

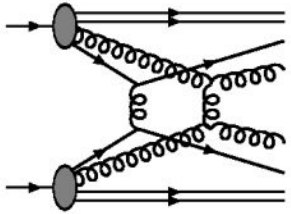
2. Resonance decays



3. Parton showers (ISR, FSR)

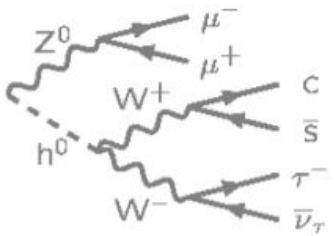


4. Multiparton interactions

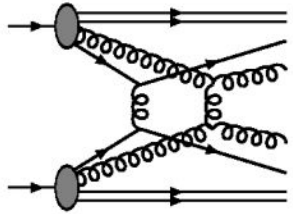


Physics processes in PYTHIA

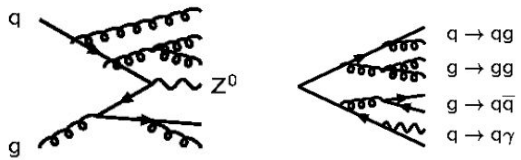
2. Resonance decays



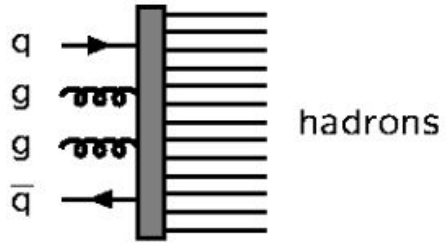
4. Multiparton interactions



3. Parton showers (ISR, FSR)

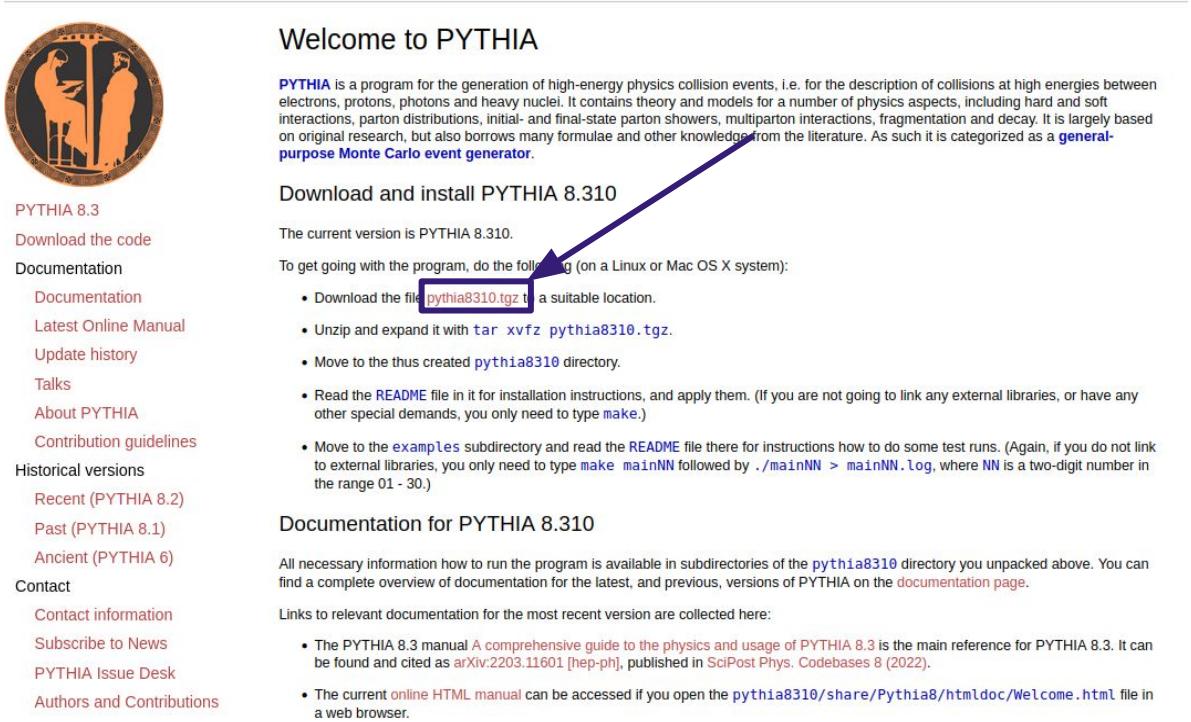


5. Hadronization



Installation

- Go to https://pythia.org/ ;
- Download pythia8310.tgz;
- move pythia8310.tgz to your installation directory;
- tar -xvfz pythia8310.tgz ;
- cd pythia8310;
- ./configure
- make



PYTHIA 8.3
Download the code
Documentation
Documentation
Latest Online Manual
Update history
Talks
About PYTHIA
Contribution guidelines
Historical versions
Recent (PYTHIA 8.2)
Past (PYTHIA 8.1)
Ancient (PYTHIA 6)
Contact
Contact information
Subscribe to News
PYTHIA Issue Desk
Authors and Contributions

Welcome to PYTHIA

PYTHIA is a program for the generation of high-energy physics collision events, i.e. for the description of collisions at high energies between electrons, protons, photons and heavy nuclei. It contains theory and models for a number of physics aspects, including hard and soft interactions, parton distributions, initial- and final-state parton showers, multiparton interactions, fragmentation and decay. It is largely based on original research, but also borrows many formulae and other knowledge from the literature. As such it is categorized as a **general-purpose Monte Carlo event generator**.

Download and install PYTHIA 8.310

The current version is PYTHIA 8.310.

To get going with the program, do the following (on a Linux or Mac OS X system):

- Download the file pythia8310.tgz to a suitable location.
- Unzip and expand it with `tar xvfz pythia8310.tgz`.
- Move to the thus created `pythia8310` directory.
- Read the `README` file in it for installation instructions, and apply them. (If you are not going to link any external libraries, or have any other special demands, you only need to type `make`.)
- Move to the `examples` subdirectory and read the `README` file there for instructions how to do some test runs. (Again, if you do not link to external libraries, you only need to type `make mainNN` followed by `./mainNN > mainNN.log`, where `NN` is a two-digit number in the range 01 - 30.)

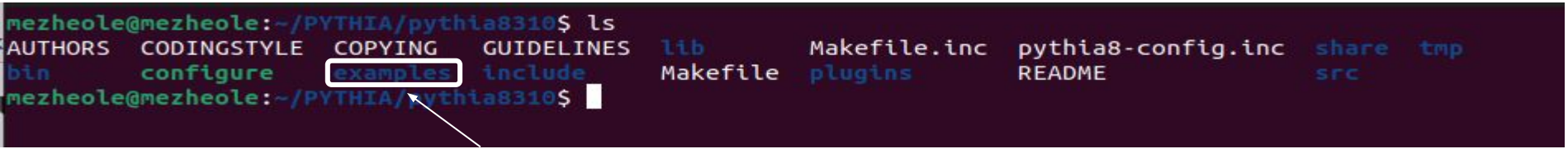
Documentation for PYTHIA 8.310

All necessary information how to run the program is available in subdirectories of the `pythia8310` directory you unpacked above. You can find a complete overview of documentation for the latest, and previous, versions of PYTHIA on the documentation page.

Links to relevant documentation for the most recent version are collected here:

- The PYTHIA 8.3 manual A comprehensive guide to the physics and usage of PYTHIA 8.3 is the main reference for PYTHIA 8.3. It can be found and cited as arXiv:2203.11601 [hep-ph], published in SciPost Phys. Codebases 8 (2022).
- The current online HTML manual can be accessed if you open the `pythia8310/share/Pythia8/htmldoc/Welcome.html` file in a web browser.

Installation



Installation

```
mezheole@mezheole:~/PYTHIA/pythia8310$ ls
AUTHORS   CODINGSTYLE   COPYING    GUIDELINES  lib       Makefile.inc  pythia8-config.inc  share  tmp
bin       configure     examples   include     Makefile  plugins       README              src
mezheole@mezheole:~/PYTHIA/pythia8310$
```

➔ cd examples

```
mezheole@mezheole:~/PYTHIA/pythia8310/examples$ ls
Bottom.cmnd          main102.cc   main154.cc   main183.cc   main25.lhe       main33.cmnd  main45.cc   main69.cc   main83.cmnd       main93.cmnd                  StUpsCandidate.h
BottomSim            main103.cc   main154.cmnd main184.cc   main26.cc        main33.pwhg  main46.cc   main70.cc   main84.cc         main93.h                     ttbar2.lhe
BottomSim.cc         main103.cmnd main155.cc   main185.cc   main27.cc        main34.cc    main46.cmnd main71.cc   main84.cmnd       main93LinkDef.h              ttbar.hdf5
main01.cc            main10.cc    main155.cmnd main18.cc    main280.cc       main34.py    main48.cc   main72.cc   main85.cc         main94.cc                    ttbar.lhe
main01.py            main10.py    main156.cc   main19.cc    main280.cmnd     main35.cc    main48.dec  main73.cc   main85.cmnd       main95.cc                    ttbar.lhe.gz
main02.cc            main111.cc   main157.cc   main200.cc   main28.cc        main36.cc    main51.cc   main74.cc   main86.cc         Makefile                     wbj_lhef3.lhe
main03.cc            main112.cc   main158.cc   main200.cmnd main29.cc        main37.cc    main52.cc   main75.cc   main86.cmnd       Makefile.inc                 w+_production_lhc_0.lhe
main03.cmnd          main113.cc   main158.cmnd main201.cc   main300.cc       main38.cc    main53.cc   main75.cmnd main87.cc         photoninproton.lhe           w+_production_lhc_1.lhe
main04.cc            main11.cc    main15.cc    main202.cc   main300.cmnd     main39.py    main53.cmnd main76.cc   main87.cmnd       powheg-dijets.lhe            w+_production_lhc_2.lhe
main04.cmnd          main121.cc   main161.cc   main202.cmnd main30.cc        main39.pyc   main54.cc   main76.cmnd main88.cc         powheg-hvq.lhe               w_production_powheg_0.lhe
main04_photons.cmnd  main121.cmnd main162.cc   main203.cc   main30.cmnd      main40.cc    main55.cc   main77.cc   main88.cmnd       README                       w_production_powheg_1.lhe
main05.cc            main12.cc    main162.py   main204.cc   main31.cc        main40.cmnd  main61.cc   main78.cc   main89.cc         Pythia8Plots                 w_production_tree_0.lhe
main06.cc            main13.cc    main163.cc   main20.cc    main31.cmnd      main41.cc    main62.cc   main80.cc   main89ckkwl.cmnd  slha1-example.spc            w_production_tree_1.lhe
main07.cc            main13.cmnd  main16.cc    main21.cc    main32.cc        main42.cc    main62.cmnd main80.cmnd main89fxfx.cmnd   slha2-example.spc            w_production_tree_2.lhe
main07.cmnd          main14.cc    main16.cmnd  main22.cc    main32.cmnd      main42.cmnd  main62.lhe  main81.cc   main89mlm.cmnd    sps1aNarrowStopGluinoRPV.spc zProduction_Ckkwl_012.lhe.gz
main08.cc            main151.cc   main171.cc   main23.cc    main32.unw       main43.cc    main63.cc   main81.cmnd main89umeps.cmnd  sps1aNarrowStopGluino.spc    zProduction_Fxfx_01.lhe.gz
main08.cmnd          main152.cc   main17.cc    main24.cc    main32_unw.par   main43.cmnd  main63.cmnd main82.cc   main89unlops.cmnd sps1aWithDecays.spc          zProduction_Mlm_012.lhe.gz
main09.cc            main153.cc   main181.cc   main24.cmnd  main333.cc       main44.cc    main64.cc   main82.cmnd main91.cc         StUpsCandDict.h              zProduction_UnlopsLoop_01.lhe.gz
main101.cc           main153.cmnd main182.cc   main25.cc    main33.cc        main44.cmnd  main68.cc   main83.cc   main93.cc         StUpsCandidate.cc            zProduction_UnlopsTree_12.lhe.gz
mezheole@mezheole:~/PYTHIA/pythia8310/examples$
```

Run

➔ make main01

Run

➔ make main01

```
mezheole@mezheole:~/PYTHIA/pythia8310/examples$ ls main*
main01              main08.cmnd   main12.cc     main158.cc    main184.cc    main23.cc     main30.cmnd   main36.cc     main45.cc     main62.lhe    main76.cmnd   main85.cmnd         main93.cc
main01.cc           main09.cc     main13.cc     main158.cmnd  main185.cc    main24.cc     main31.cc     main37.cc     main46.cc     main63.cc     main77.cc     main86.cc           main93.cmnd
main01.py           main101.cc    main13.cmnd   main15.cc     main18.cc     main24.cmnd   main31.cmnd   main38.cc     main46.cmnd   main63.cmnd   main78.cc     main86.cmnd         main93.h
main02.cc           main102.cc    main14.cc     main161.cc    main19.cc     main25.cc     main32.cc     main39.py     main48.cc     main64.cc     main80.cc     main87.cc           main93LinkDef.h
main03.cc           main103.cc    main151.cc    main162.cc    main200.cc    main25.lhe    main32.cmnd   main39.pyc    main48.dec    main68.cc     main80.cmnd   main87.cmnd         main94.cc
main03.cmnd         main103.cmnd  main152.cc    main162.py    main200.cmnd  main26.cc     main32.unw    main40.cc     main51.cc     main69.cc     main81.cc     main88.cc           main95.cc
main04.cc           main10.cc     main153.cc    main163.cc    main201.cc    main27.cc     main32_unw.par main40.cmnd  main52.cc     main70.cc     main81.cmnd   main88.cmnd
main04.cmnd         main10.py     main153.cmnd  main16.cc     main202.cc    main280.cc    main333.cc    main41.cc     main53.cc     main71.cc     main82.cc     main89.cc
main04_photons.cmnd main111.cc    main154.cc    main16.cmnd   main202.cmnd  main280.cmnd  main33.cc     main42.cc     main53.cmnd   main72.cc     main82.cmnd   main89ckkwl.cmnd
main05.cc           main112.cc    main154.cmnd  main171.cc    main203.cc    main28.cc     main33.cmnd   main42.cmnd   main54.cc     main73.cc     main83.cc     main89fxfx.cmnd
main06.cc           main113.cc    main155.cc    main17.cc     main204.cc    main29.cc     main33.pwhg   main43.cc     main55.cc     main74.cc     main83.cmnd   main89mlm.cmnd
main07.cc           main11.cc     main155.cmnd  main181.cc    main20.cc     main300.cc    main34.cc     main43.cmnd   main61.cc     main75.cc     main84.cc     main89umeps.cmnd
main07.cmnd         main121.cc    main156.cc    main182.cc    main21.cc     main300.cmnd  main34.py     main44.cc     main62.cc     main75.cmnd   main84.cmnd   main89unlops.cmnd
main08.cc           main121.cmnd  main157.cc    main183.cc    main22.cc     main30.cc     main35.cc     main44.cmnd   main62.cmnd   main76.cc     main85.cc     main91.cc
mezheole@mezheole:~/PYTHIA/pythia8310/examples$
```

➔ ./main01

Run

➔ make main01

```
mezheole@mezheole:~/PYTHIA/pythia8310/examples$ ls main*
main01              main08.cmnd   main12.cc     main158.cc    main184.cc    main23.cc     main30.cmnd   main36.cc     main45.cc     main62.lhe    main76.cmnd   main85.cmnd         main93.cc
main01.cc           main09.cc     main13.cc     main158.cmnd  main185.cc    main24.cc     main31.cc     main37.cc     main46.cc     main63.cc     main77.cc     main86.cc           main93.cmnd
main01.py           main101.cc    main13.cmnd   main15.cc     main18.cc     main24.cmnd   main31.cmnd   main38.cc     main46.cmnd   main63.cmnd   main78.cc     main86.cmnd         main93.h
main02.cc           main102.cc    main14.cc     main161.cc    main19.cc     main25.cc     main32.cc     main39.py     main48.cc     main64.cc     main80.cc     main87.cc           main93LinkDef.h
main03.cc           main103.cc    main151.cc    main162.cc    main200.cc    main25.lhe    main32.cmnd   main39.pyc    main48.dec    main68.cc     main80.cmnd   main87.cmnd         main94.cc
main03.cmnd         main103.cmnd  main152.cc    main162.py    main200.cmnd  main26.cc     main32.unw    main40.cc     main51.cc     main69.cc     main81.cc     main88.cc           main95.cc
main04.cc           main10.cc     main153.cc    main163.cc    main201.cc    main27.cc     main32_unw.par main40.cmnd  main52.cc     main70.cc     main81.cmnd   main88.cmnd
main04.cmnd         main10.py     main153.cmnd  main16.cc     main202.cc    main280.cc    main333.cc    main41.cc     main53.cc     main71.cc     main82.cc     main89.cc
main04_photons.cmnd main111.cc    main154.cc    main16.cmnd   main202.cmnd  main280.cmnd  main33.cc     main42.cc     main53.cmnd   main72.cc     main82.cmnd   main89ckkwl.cmnd
main05.cc           main112.cc    main154.cmnd  main171.cc    main203.cc    main28.cc     main33.cmnd   main42.cmnd   main54.cc     main73.cc     main83.cc     main89fxfx.cmnd
main06.cc           main113.cc    main155.cc    main17.cc     main204.cc    main29.cc     main33.pwhg   main43.cc     main55.cc     main74.cc     main83.cmnd   main89mlm.cmnd
main07.cc           main11.cc     main155.cmnd  main181.cc    main20.cc     main300.cc    main34.cc     main43.cmnd   main61.cc     main75.cc     main84.cc     main89umeps.cmnd
main07.cmnd         main121.cc    main156.cc    main182.cc    main21.cc     main300.cmnd  main34.py     main44.cc     main62.cc     main75.cmnd   main84.cmnd   main89unlops.cmnd
main08.cc           main121.cmnd  main157.cc    main183.cc    main22.cc     main30.cc     main35.cc     main44.cmnd   main62.cmnd   main76.cc     main85.cc     main91.cc
mezheole@mezheole:~/PYTHIA/pythia8310/examples$
```

➔ ./main01

Compile and run PYTHIA programs outside the examples/ directory:

export PYTHIA8PATH = <set to head Pythia directory>
export PYTHIA8DATA = $PYTHIA8PATH/share/Pythia8/xmldoc
export LD_LIBRARY_PATH = $PYTHIA8PATH/lib:$LD_LIBRARY_PATH

Program structure

- Proper header file must be included:

```
#include "Pythia8/Pythia.h"
using namespace Pythia8;
```

- create a generator object:

```
Pythia pythia;
```

- Pythia's settings and particle data:

```
pythia.readString(string);
pythia.readFile(fileName);
```

- initialize all aspects of the subsequent generation:

```
pythia.init();
```

- to generate the next event

```
pythia.next();
```

- run statistics

```
pythia.stat();
```

Settings

Input strings for changing settings: ***settingGroup:nameOfSetting = value***

For example,

ProcessLevel:ISR = off
ProcessLevel:FSR = on
ProcessLevel:MPI =on

Settings

Input strings for changing settings: ***settingGroup:nameOfSetting = value***

For example,

ProcessLevel:ISR = off
ProcessLevel:FSR = on
ProcessLevel:MPI =on

PYTHIA 8.3 supports four different types of settings:

- **flag** is a boolean true or false.

  Acceptable input alternatives include on/off, yes/no, and 1/0

Settings

Input strings for changing settings: ***settingGroup:nameOfSetting = value***

For example,

ProcessLevel:ISR = off
ProcessLevel:FSR = on
ProcessLevel:MPI =on

PYTHIA 8.3 supports four different types of settings:

- **flag** is a boolean true or false.

  Acceptable input alternatives include on/off, yes/no, and 1/0
- **mode** is an integer switch enumerating either available options or a wider range of values.

  Acceptable values are integers

Settings

Input strings for changing settings: ***settingGroup:nameOfSetting = value***

For example, ***ProcessLevel:ISR = off***
ProcessLevel:FSR = on
ProcessLevel:MPI =on

PYTHIA 8.3 supports four different types of settings:

- **flag** is a boolean true or false

  Acceptable input alternatives include on/off, yes/no, and 1/0
- **mode** is an integer switch enumerating either available options or a wider range of values.

  Acceptable values are integers
- **parm** is a real number parameter

Settings

Input strings for changing settings: ***settingGroup:nameOfSetting = value***

For example,

ProcessLevel:ISR = off
ProcessLevel:FSR = on
ProcessLevel:MPI =on

PYTHIA 8.3 supports four different types of settings:

- **flag** is a boolean true or false

  Acceptable input alternatives include on/off, yes/no, and 1/0
- **mode** is an integer switch enumerating either available options or a wider range of values

  Acceptable values are integers
- **parm** is a real number parameter
- **word** is a character string (***PDF:pSet = LHAPDF6:MRSTMCal/0***)
  It cannot contain single or double quotation marks, or curly braces, *i.e. { }*

Settings

The user can read in settings in one of two ways:

- ***pythia.readString()*** inside the code;
- ***pythia.readFile("cardfile.cmnd").***

not requiring a recompilation every time

Settings: Beams

Beams:idA =incoming particle 1

Beams:idB = incoming particle 2

PDG codes: ***MONTE CARLO PARTICLE NUMBERING SCHEME***

Settings: Beams

Beams:idA =incoming particle 1

Beams:idB = incoming particle 2

PDG codes: ***MONTE CARLO PARTICLE NUMBERING SCHEME***

***pp* collision(default)**: *Beams:idA = 2212*
Beams:idB = 2212

***p$\bar{p}$* collision**: *Beams:idA = 2212*
Beams:idB = -2212

Settings: Beams

Beams:idA =incoming particle 1

Beams:idB = incoming particle 2

PDG codes: *MONTE CARLO PARTICLE NUMBERING SCHEME*

***pp* collision(default)**: *Beams:idA = 2212* *Beams:idB = 2212*

$p\bar{p}$ collision: *Beams:idA = 2212* *Beams:idB = -2212*

e^+e^-collision: *Beams:idA = 11* *Beams:idB = -11*

Settings: Beams

Beams:idA =incoming particle 1

Beams:idB = incoming particle 2

PDG codes: *MONTE CARLO PARTICLE NUMBERING SCHEME*

***pp* collision(default)**: *Beams:idA = 2212* *Beams:idB = 2212*

$p\bar{p}$ collision: *Beams:idA = 2212* *Beams:idB = -2212*

e^+e^-collision: *Beams:idA = 11* *Beams:idB = -11*

Currently available beams include:

- ❑ protons (2212)
- ❑ neutrons (2112)
- ❑ pions (±211, 111)
- ❑ most other light hadrons
- ❑ electrons (11)
- ❑ muons (13)
- ❑ photons (22)
- ❑ several heavy-ion species.

Settings: Beams

Beams:idA =incoming particle 1

Beams:idB = incoming particle 2

PDG codes: *MONTE CARLO PARTICLE NUMBERING SCHEME*

***pp* collision(default)**: *Beams:idA = 2212* *Beams:idB = 2212*

$p\bar{p}$ collision: *Beams:idA = 2212* *Beams:idB = -2212*

e^+e^-collision: *Beams:idA = 11* *Beams:idB = -11*

Currently available beams include:

- ❑ protons (2212)
- ❑ neutrons (2112)
- ❑ pions (±211, 111)
- ❑ most other light hadrons
- ❑ electrons (11)
- ❑ muons (13)
- ❑ photons (22)
- ❑ several heavy-ions pecies

Collision energy: *Beams:eCM = 200*

Units of GeV in CM frame

Settings: Beams

Beams:idA =incoming particle 1

Beams:idB = incoming particle 2

PDG codes: *MONTE CARLO PARTICLE NUMBERING SCHEME*

***pp* collision(default)**: *Beams:idA = 2212* *Beams:idB = 2212*

$p\bar{p}$ collision: *Beams:idA = 2212* *Beams:idB = -2212*

e^+e^-collision: *Beams:idA = 11* *Beams:idB = -11*

Currently available beams include:

- ❑ protons (2212)
- ❑ neutrons (2112)
- ❑ pions (±211, 111)
- ❑ most other light hadrons
- ❑ electrons (11)
- ❑ muons (13)
- ❑ photons (22)
- ❑ several heavy-ionspecies.

Collision energy: *Beams:eCM = 200*

Units of GeV in CM frame

Other options:

Beams:frameType = 2
Beams:idA = 2212
Beams:eA = 920.
Beams:idB = -11
Beams:eB = 27.5

Analysis of generated event: The Vec4 class

A generated "event" is essentially a list of particles — initial, final, or intermediate — that are generated sequentially based on probabilistic calculations.

The Vec4 class (*https://pythia.org/latest-manual/FourVectors.html*)

- ***px(), py(), pz(), e()*** - access the individual components
- ***mCalc()*** - calculated mass $\sqrt{E^2 - p_x^2 - p_y^2 - p_z^2}$
- ***pT()*** and ***pAbs()*** - the transverse momentum and the absolute value of the three-momentum
- ***theta()***, ***eta()***, ***phi()*** - the polar and azimuthal angles, rapidity and pseudorapidity
- ***rot(double theta, double phi)*** to rotate the three-momentum.
- ***bst(const Vec4& p)*** and ***bstback(const Vec4& p)*** to boost the current vector by $\vec{\beta} = \pm\frac{\vec{p}}{E}$

Analysis of generated event: The Particle class

The Particle class (*https://pythia.org/latest-manual/ParticleProperties.html*)

The Particle class forms the fundamental particle unit

Properties:

- ***id()*** for the PDG code;
- ***status()*** for the status of the current particle (***isFinal()***);
- ***p()*** returns a four-vector whereas ***px()***, ***py()***, ***pz()***, ***e()*** - directly to access components;
- ***mother1()***, ***mother2()*** refer to the indices of the first and last mother;
- ***motherList()*** returns a vector of all the mother indices;
- ***daughter1()***, ***daughter2()*** refer to the indices of the first and the last daughter;
- ***daughterList()*** returns a vector of all the daughter indices.

Analysis of generated event: The Event class

The Event class (*https://pythia.org/latest-manual/EventRecord.html*)

It contains a dynamic array (vector) of particles along with helper methods that are useful to extract information from the array.

- The individual particles can be accessed simply by using their index in the event (e.g. ***pythia.event[i]***).
- All methods corresponding to the particle then can be accessed:
  - ***pythia.event[i].px()***
  - ***pythia.event[i].py()***
  - ***pythia.event[i].pz()***
  - ***pythia.event[i].pt()***
  - ***pythia.event[i].phi()***
  - ***pythia.event[i].theta()***

Program output

- ***pythia.event.list()*** - provides the main properties of each particles

Program output

- ***pythia.event.list()*** - provides the main properties of each particles
- ***pythia.stat()*** - printout of the statistics, i.e. number of tried and accepted events, as well as the number of events produced for each process and the resulting cross

Program output

- ***pythia.event.list()*** - listing of the full event
- ***pythia.stat()*** - printout of the statistics, i.e. number of tried and accepted events, as well as the number of events produced for each process and the resulting cross

Pythia also provides rudimentary built-in histogramming via the **Hist class**.

Methods :

- ***Hist(string title, int numberOfBins, double xMin, double xMax)*** the constructor, defining a histogram.

Program output

- ***pythia.event.list()*** - listing of the full event
- ***pythia.stat()*** - printout of the statistics, i.e. number of tried and accepted events, as well as the number of events produced for each process and the resulting cross

Pythia also provides rudimentary built-in histogramming via the **Hist class**.

Methods :

- ***Hist(string title, int numberOfBins, double xMin, double xMax)*** the constructor, defining a histogram.

- ***fill(double value, double weight = 1.0)*** to fill the histogram with an optional weight.

Program output

- ***pythia.event.list()*** - listing of the full event
- ***pythia.stat()*** - printout of the statistics, i.e. number of tried and accepted events, as well as the number of events produced for each process and the resulting cross

Pythia also provides rudimentary built-in histogramming via the **Hist class**.

Methods :

- ***Hist(string title, int numberOfBins, double xMin, double xMax)*** the constructor, defining a histogram.
- ***fill(double value, double weight = 1.0)*** to fill the histogram with an optional weight.
- ***std::cout < < myhist*** to output the histogram.

main01

```
#include "Pythia8/Pythia.h"
using namespace Pythia8;
int main() {
  // Generator. Process selection. LHC initialization. Histogram.
  Pythia pythia;
  pythia.readString("Beams:eCM = 8000.");
  pythia.readString("HardQCD:all = on");
  pythia.readString("PhaseSpace:pTHatMin = 20.");
  pythia.init();
  Hist mult("charged multiplicity", 100, -0.5, 799.5);
  // Begin event loop. Generate event. Skip if error. List first one.
  for (int iEvent = 0; iEvent < 100; ++iEvent) {
          if (!pythia.next()) continue;
          // Find number of all final charged particles and fill histogram.
          int nCharged = 0;
          for (int i = 0; i < pythia.event.size(); ++i)
          if (pythia.event[i].isFinal() && pythia.event[i].isCharged())
          ++nCharged;
          mult.fill( nCharged );
  // End of event loop. Statistics. Histogram. Done.
  }
  pythia.stat();
  cout << mult;
  return 0;
```

main01

```
#include "Pythia8/Pythia.h"
using namespace Pythia8;
int main() {
  // Generator. Process selection. LHC initialization. Histogram.
  Pythia pythia;
  pythia.readString("Beams:eCM = 8000.");
  pythia.readString("HardQCD:all = on");
  pythia.readString("PhaseSpace:pTHatMin = 20.");
  pythia.init();
  Hist mult("charged multiplicity", 100, -0.5, 799.5);
  // Begin event loop. Generate event. Skip if error. List first one.
  for (int iEvent = 0; iEvent < 100; ++iEvent) {
          if (!pythia.next()) continue;
          // Find number of all final charged particles and fill histogram.
          int nCharged = 0;
          for (int i = 0; i < pythia.event.size(); ++i)
          if (pythia.event[i].isFinal() && pythia.event[i].isCharged())
          ++nCharged;
          mult.fill( nCharged );
  // End of event loop. Statistics. Histogram. Done.
  }
  pythia.stat();
  cout << mult;
  return 0;
```

main01

```
#include "Pythia8/Pythia.h"
using namespace Pythia8;
int main() {
  // Generator. Process selection. LHC initialization. Histogram.
  Pythia pythia;
  pythia.readString("Beams:eCM = 8000.");
  pythia.readString("HardQCD:all = on");
  pythia.readString("PhaseSpace:pTHatMin = 20.");
  pythia.init();
  Hist mult("charged multiplicity", 100, -0.5, 799.5);
  // Begin event loop. Generate event. Skip if error. List first one.
  for (int iEvent = 0; iEvent < 100; ++iEvent) {
          if (!pythia.next()) continue;
          // Find number of all final charged particles and fill histogram.
          int nCharged = 0;
          for (int i = 0; i < pythia.event.size(); ++i)
          if (pythia.event[i].isFinal() && pythia.event[i].isCharged())
          ++nCharged;
          mult.fill( nCharged );
  // End of event loop. Statistics. Histogram. Done.
  }
  pythia.stat();
  cout << mult;
  return 0;
```

main01

```
#include "Pythia8/Pythia.h"
using namespace Pythia8;
int main() {
  // Generator. Process selection. LHC initialization. Histogram.
  Pythia pythia;
  pythia.readString("Beams:eCM = 8000.");
  pythia.readString("HardQCD:all = on");
  pythia.readString("PhaseSpace:pTHatMin = 20.");
  pythia.init();
  Hist mult("charged multiplicity", 100, -0.5, 799.5);
  // Begin event loop. Generate event. Skip if error. List first one.
  for (int iEvent = 0; iEvent < 100; ++iEvent) {
          if (!pythia.next()) continue;
          // Find number of all final charged particles and fill histogram.
          int nCharged = 0;
          for (int i = 0; i < pythia.event.size(); ++i)
          if (pythia.event[i].isFinal() && pythia.event[i].isCharged())
          ++nCharged;
          mult.fill( nCharged );
  // End of event loop. Statistics. Histogram. Done.
  }
  pythia.stat();
  cout << mult;
  return 0;
```

main01

```
#include "Pythia8/Pythia.h"
using namespace Pythia8;
int main() {
  // Generator. Process selection. LHC initialization. Histogram.
  Pythia pythia;
  pythia.readString("Beams:eCM = 8000.");
  pythia.readString("HardQCD:all = on");
  pythia.readString("PhaseSpace:pTHatMin = 20.");
  pythia.init();
  Hist mult("charged multiplicity", 100, -0.5, 799.5);
  // Begin event loop. Generate event. Skip if error. List first one.
  for (int iEvent = 0; iEvent < 100; ++iEvent) {
          if (!pythia.next()) continue;
          // Find number of all final charged particles and fill histogram.
          int nCharged = 0;
          for (int i = 0; i < pythia.event.size(); ++i)
          if (pythia.event[i].isFinal() && pythia.event[i].isCharged())
          ++nCharged;
          mult.fill( nCharged );
  // End of event loop. Statistics. Histogram. Done.
  }
  pythia.stat();
  cout << mult;
  return 0;
```

main01

```
#include "Pythia8/Pythia.h"
using namespace Pythia8;
int main() {
  // Generator. Process selection. LHC initialization. Histogram.
  Pythia pythia;
  pythia.readString("Beams:eCM = 8000.");
  pythia.readString("HardQCD:all = on");
  pythia.readString("PhaseSpace:pTHatMin = 20.");
  pythia.init();
  Hist mult("charged multiplicity", 100, -0.5, 799.5);
  // Begin event loop. Generate event. Skip if error. List first one.
  for (int iEvent = 0; iEvent < 100; ++iEvent) {
          if (!pythia.next()) continue;
          // Find number of all final charged particles and fill histogram.
          int nCharged = 0;
          for (int i = 0; i < pythia.event.size(); ++i)
          if (pythia.event[i].isFinal() && pythia.event[i].isCharged())
          ++nCharged;
          mult.fill( nCharged );
  // End of event loop. Statistics. Histogram. Done.
  }
  pythia.stat();
  cout << mult;
  return 0;
```

ensures that if an error occurs during event generation, the loop will move on to the next iteration without executing the code inside the loop body, preventing the program from crashing or producing incorrect results due to the error.

main01

```
#include "Pythia8/Pythia.h"
using namespace Pythia8;
int main() {
  // Generator. Process selection. LHC initialization. Histogram.
  Pythia pythia;
  pythia.readString("Beams:eCM = 8000.");
  pythia.readString("HardQCD:all = on");
  pythia.readString("PhaseSpace:pTHatMin = 20.");
  pythia.init();
  Hist mult("charged multiplicity", 100, -0.5, 799.5);
  // Begin event loop. Generate event. Skip if error. List first one.
  for (int iEvent = 0; iEvent < 100; ++iEvent) {
          if (!pythia.next()) continue;
          // Find number of all final charged particles and fill histogram.
          int nCharged = 0;
          for (int i = 0; i < pythia.event.size(); ++i)
          if (pythia.event[i].isFinal() && pythia.event[i].isCharged())
          ++nCharged;
          mult.fill( nCharged );
  // End of event loop. Statistics. Histogram. Done.
  }
  pythia.stat();
  cout << mult;
  return 0;
```

ensures that if an error occurs during event generation, the loop will move on to the next iteration without executing the code inside the loop body, preventing the program from crashing or producing incorrect results due to the error.

returns the number of particles present in the current event.

main01

```
#include "Pythia8/Pythia.h"
using namespace Pythia8;
int main() {
  // Generator. Process selection. LHC initialization. Histogram.
  Pythia pythia;
  pythia.readString("Beams:eCM = 8000.");
  pythia.readString("HardQCD:all = on");
  pythia.readString("PhaseSpace:pTHatMin = 20.");
  pythia.init();
  Hist mult("charged multiplicity", 100, -0.5, 799.5);
  // Begin event loop. Generate event. Skip if error. List first one.
  for (int iEvent = 0; iEvent < 100; ++iEvent) {
        if (!pythia.next()) continue;
        // Find number of all final charged particles and fill histogram.
        int nCharged = 0;
        for (int i = 0; i < pythia.event.size(); ++i)
        if (pythia.event[i].isFinal() && pythia.event[i].isCharged())
        ++nCharged;
        mult.fill( nCharged );
  // End of event loop. Statistics. Histogram. Done.
  }
  pythia.stat();
  cout << mult;
  return 0;
```

ensures that if an error occurs during event generation, the loop will move on to the next iteration without executing the code inside the loop body, preventing the program from crashing or producing incorrect results due to the error.

returns the number of particles present in the current event.

To compile
make main01

To run
./main01

Compile with ROOT

- cd pythia8310
- ./configure --with-root=root-installation-directory
- make

examples/main91.cc - example of histogramming with ROOT

main92.cc - example program showing how PYTHIA events can be stored in ROOT trees