你好: Hello
你早: Good morning

sPHENIX INTT Analysis Workshop,
November 15th, 2023
National Central University, Taiwai

Nǐ　zǎo
你早, Fun4All!

Fun4All Tutorial Part-II

G. Nukazuka (RIKEN/RBRC)

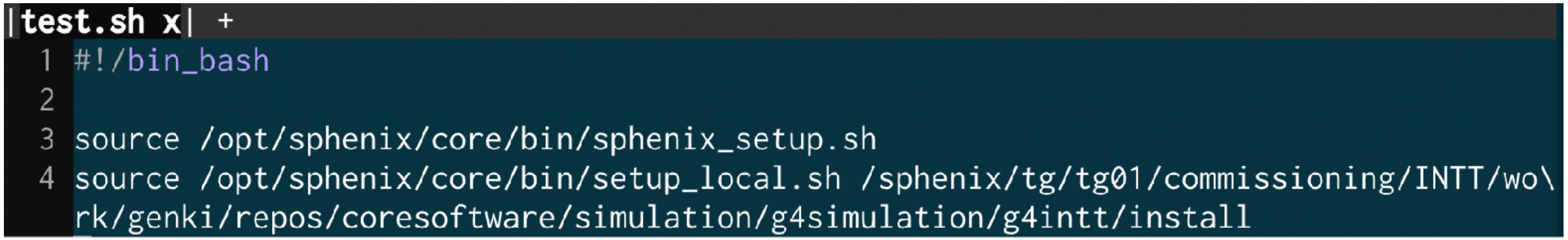
What we did yesterday

- We started from the minimum sample Fun4All_minimum.C. We ran it.
- We generated our own analysis module by using CreateSubsysRecoModule.pl.
- We compiled and installed the analysis module.
- We modified LD_LIBRARY_PATH and ROOT_INCLUDE_PATH by using /opt/sphenix/core/bin/sphenix_setup.sh
- We ran the sample macro Fun4All_minimum_2.C.

What's next? Analysis?

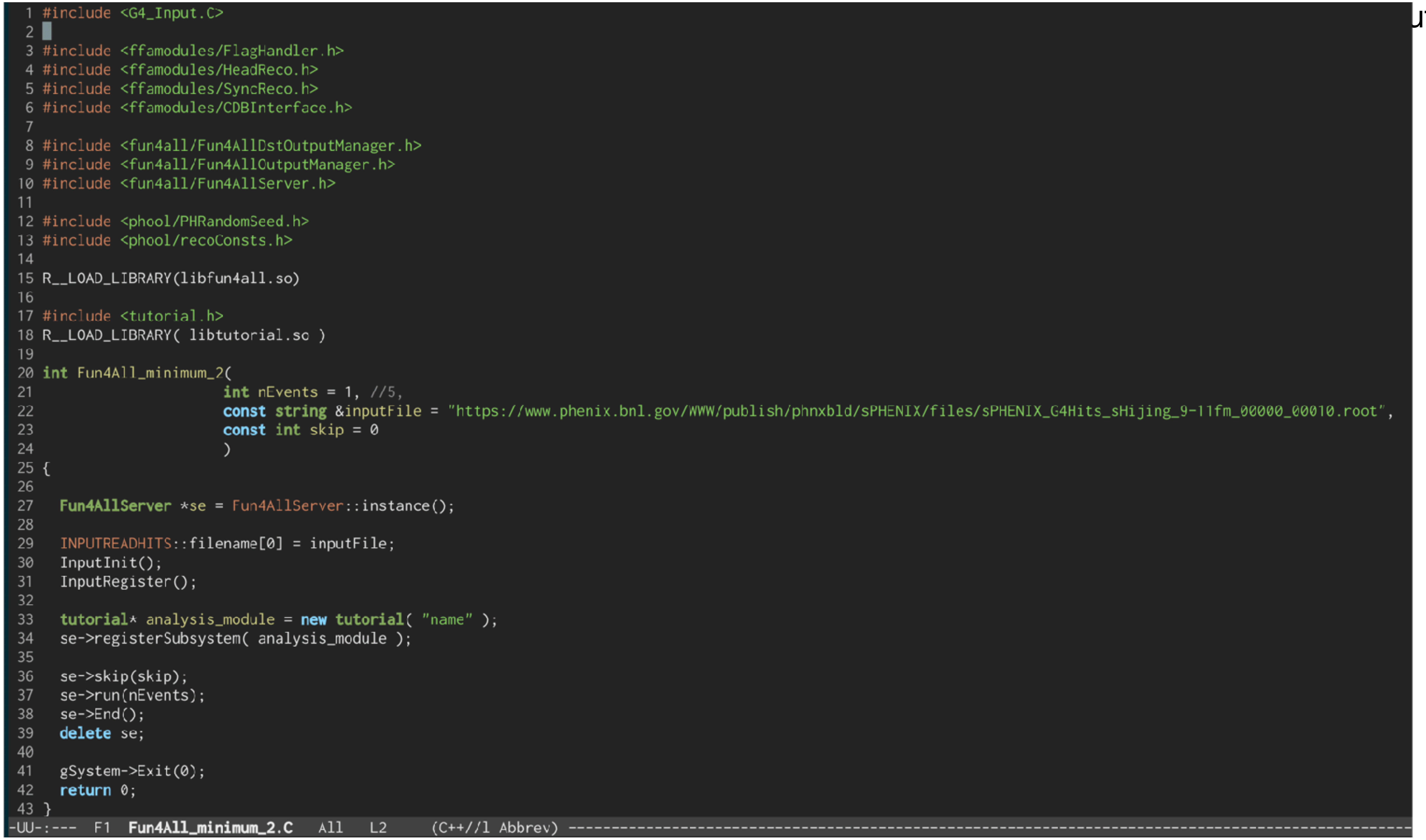
Before staring data analysis, let's input ~~a DST file.~~

I couldn't run with a DST input somehow. Let's move to MC event.



```
|test.sh x| +
#!/bin_bash

source /opt/sphenix/core/bin/sphenix_setup.sh
source /opt/sphenix/core/bin/setup_local.sh /sphenix/tg/tg01/commissioning/INTT/wo\
rk/genki/repos/coresoftware/simulation/g4simulation/g4intt/install
```

Implementation of your analysis module

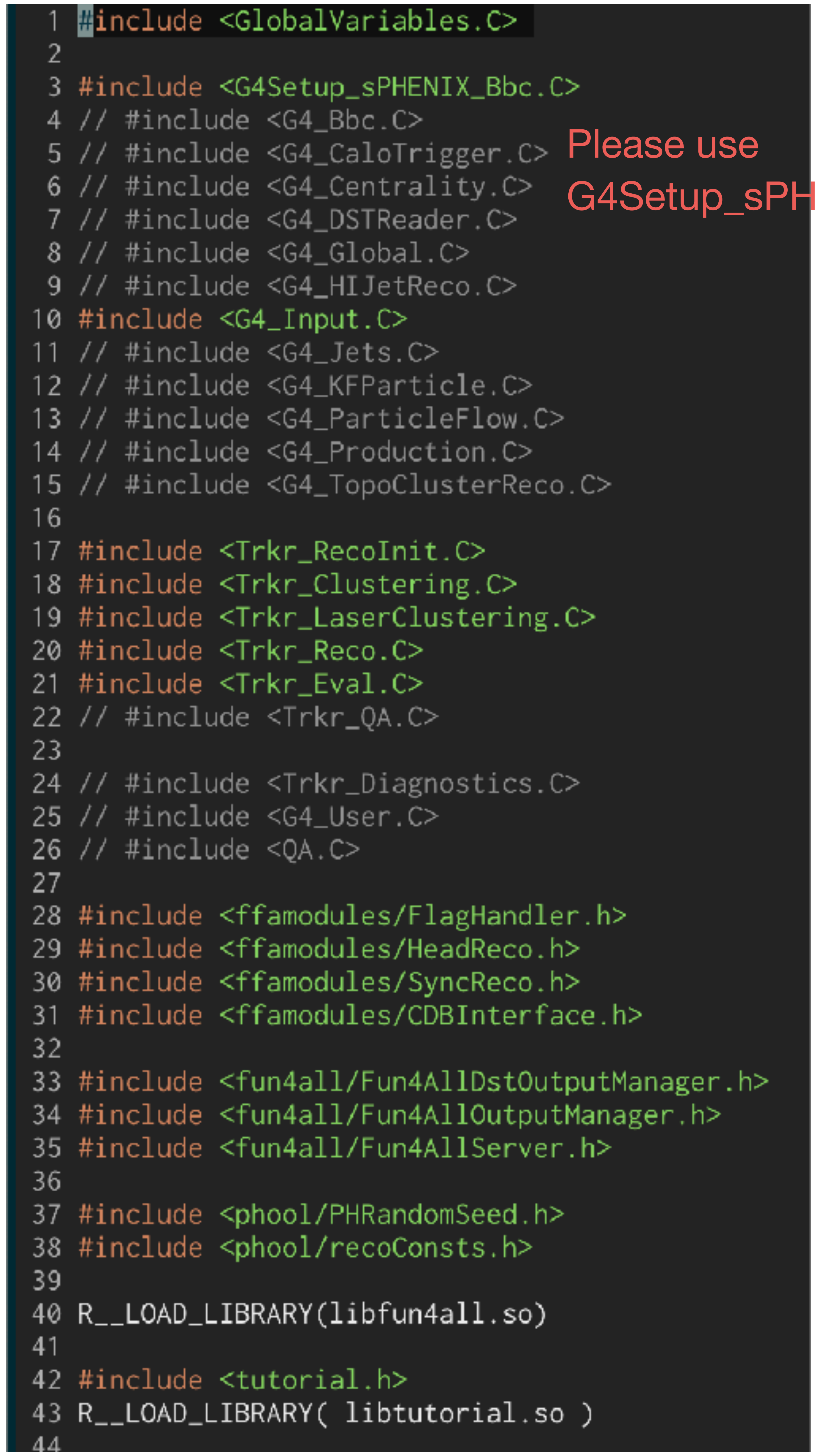


```
#include <G4_Input.C>

#include <ffamodules/FlagHandler.h>
#include <ffamodules/HeadReco.h>
#include <ffamodules/SyncReco.h>
#include <ffamodules/CDBInterface.h>

#include <fun4all/Fun4AllDstOutputManager.h>
#include <fun4all/Fun4AllOutputManager.h>
#include <fun4all/Fun4AllServer.h>

#include <phool/PHRandomSeed.h>
#include <phool/recoConsts.h>

R__LOAD_LIBRARY(libfun4all.so)

#include <tutorial.h>
R__LOAD_LIBRARY( libtutorial.so )

int Fun4All_minimum_2(
                     int nEvents = 1, //5,
                     const string &inputFile = "https://www.phenix.bnl.gov/WWW/publish/phnxbld/sPHENIX/files/sPHENIX_G4Hits_sHijing_9-11fm_00000_00010.root",
                     const int skip = 0
                     )
{

  Fun4AllServer *se = Fun4AllServer::instance();

  INPUTREADHITS::filename[0] = inputFile;
  InputInit();
  InputRegister();

  tutorial* analysis_module = new tutorial( "name" );
  se->registerSubsystem( analysis_module );

  se->skip(skip);
  se->run(nEvents);
  se->End();
  delete se;

  gSystem->Exit(0);
  return 0;
}
-UU-:---  F1  Fun4All_minimum_2.C   All   L2     (C++//l Abbrev) ------------------------------------------------
```

utorial.h, .cc

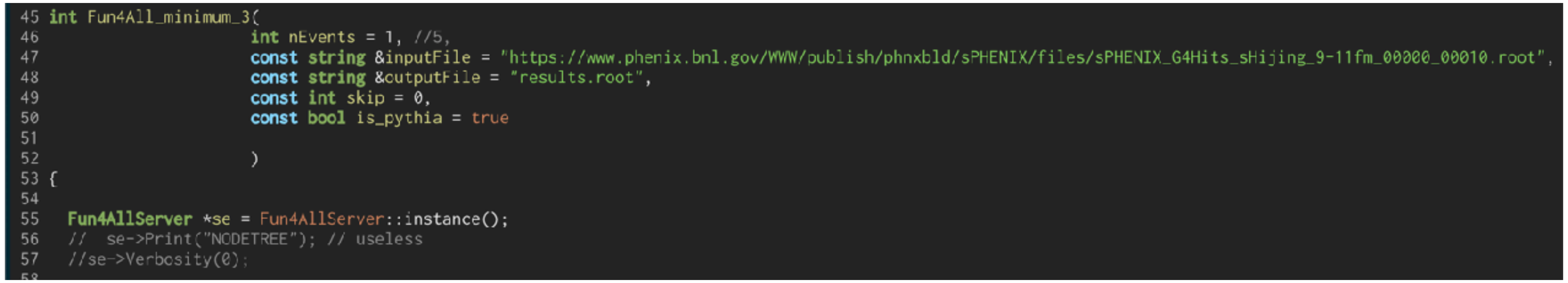
Implementation of your analysis module

ref: Fun4All_minimum_3.C



```
#include <GlobalVariables.C>

#include <G4Setup_sPHENIX_Bbc.C>
// #include <G4_Bbc.C>
// #include <G4_CaloTrigger.C>
// #include <G4_Centrality.C>
// #include <G4_DSTReader.C>
// #include <G4_Global.C>
// #include <G4_HIJetReco.C>
#include <G4_Input.C>
// #include <G4_Jets.C>
// #include <G4_KFParticle.C>
// #include <G4_ParticleFlow.C>
// #include <G4_Production.C>
// #include <G4_TopoClusterReco.C>

#include <Trkr_RecoInit.C>
#include <Trkr_Clustering.C>
#include <Trkr_LaserClustering.C>
#include <Trkr_Reco.C>
#include <Trkr_Eval.C>
// #include <Trkr_QA.C>

// #include <Trkr_Diagnostics.C>
// #include <G4_User.C>
// #include <QA.C>

#include <ffamodules/FlagHandler.h>
#include <ffamodules/HeadReco.h>
#include <ffamodules/SyncReco.h>
#include <ffamodules/CDBInterface.h>

#include <fun4all/Fun4AllDstOutputManager.h>
#include <fun4all/Fun4AllOutputManager.h>
#include <fun4all/Fun4AllServer.h>

#include <phool/PHRandomSeed.h>
#include <phool/recoConsts.h>

R__LOAD_LIBRARY(libfun4all.so)

#include <tutorial.h>
R__LOAD_LIBRARY( libtutorial.so )
```

Please use G4Setup_sPHENIX.C.



```
int Fun4All_minimum_3(
                        int nEvents = 1, //5,
                        const string &inputFile = "https://www.phenix.bnl.gov/WWW/publish/phnxbld/sPHENIX/files/sPHENIX_G4Hits_sHijing_9-11fm_00000_00010.root",
                        const string &outputFile = "results.root",
                        const int skip = 0,
                        const bool is_pythia = true

                        )
{

  Fun4AllServer *se = Fun4AllServer::instance();
  //  se->Print("NODETREE"); // useless
  //se->Verbosity(0);
```

The amount of code in Fun4All_minimum_3.C is drastically changed from _2.C (44 → 381 lines). That's because

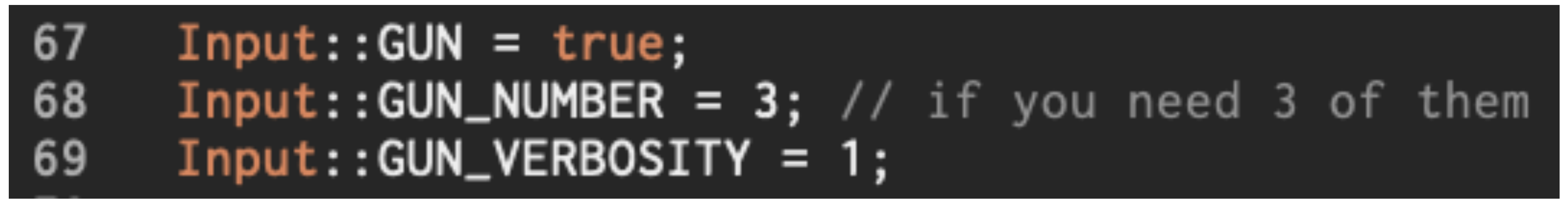
- Configuration of event generator
- Geometry configuration

Let's check them.

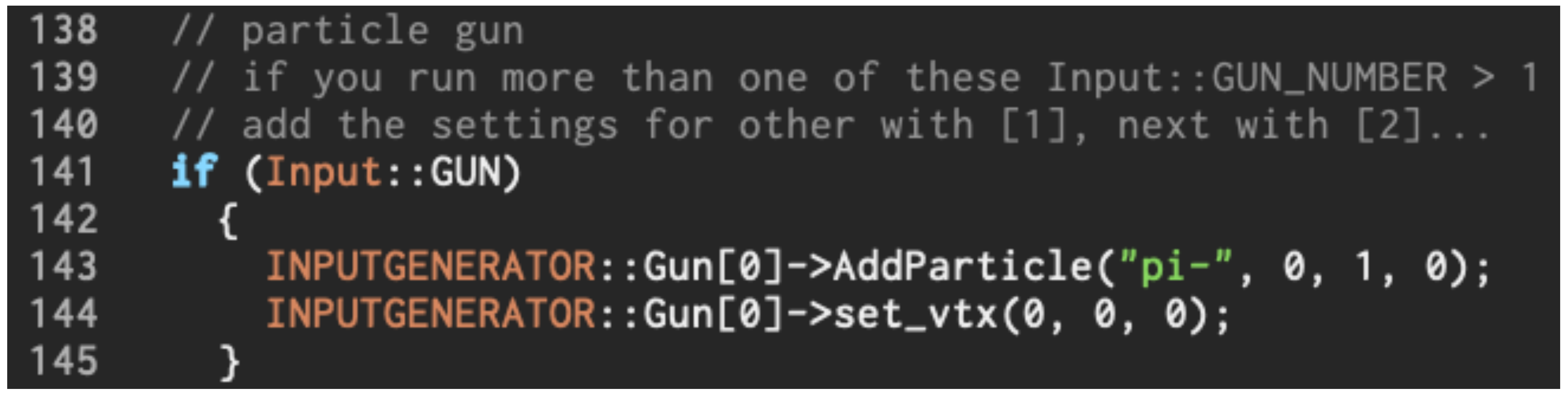
Practical example2: MC, Event generator

Event generators:

- GUN: A particle gun to shoot particles as you want
- SIMPLE: A particle gun with some realistic kinematics?
- Pythia6: General event generator
- Pythia8: General event generator
- DZERO: D_0 generator
- LAMBDAC: Λ_c generator (not ready)
- UPSILON: Υ generator
- HEPMC: ?



```
Input::GUN = true;
Input::GUN_NUMBER = 3; // if you need 3 of them
Input::GUN_VERBOSITY = 1;
```



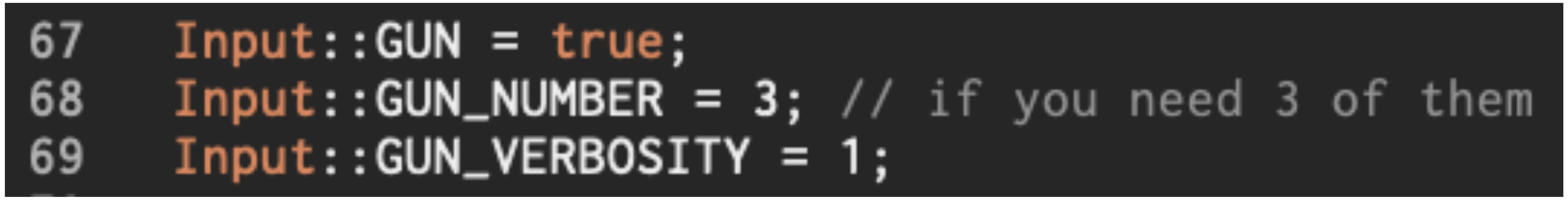
```
 // particle gun
 // if you run more than one of these Input::GUN_NUMBER > 1
 // add the settings for other with [1], next with [2]...
 if (Input::GUN)
   {
     INPUTGENERATOR::Gun[0]->AddParticle("pi-", 0, 1, 0);
     INPUTGENERATOR::Gun[0]->set_vtx(0, 0, 0);
   }
```

Configuration for GUN generator

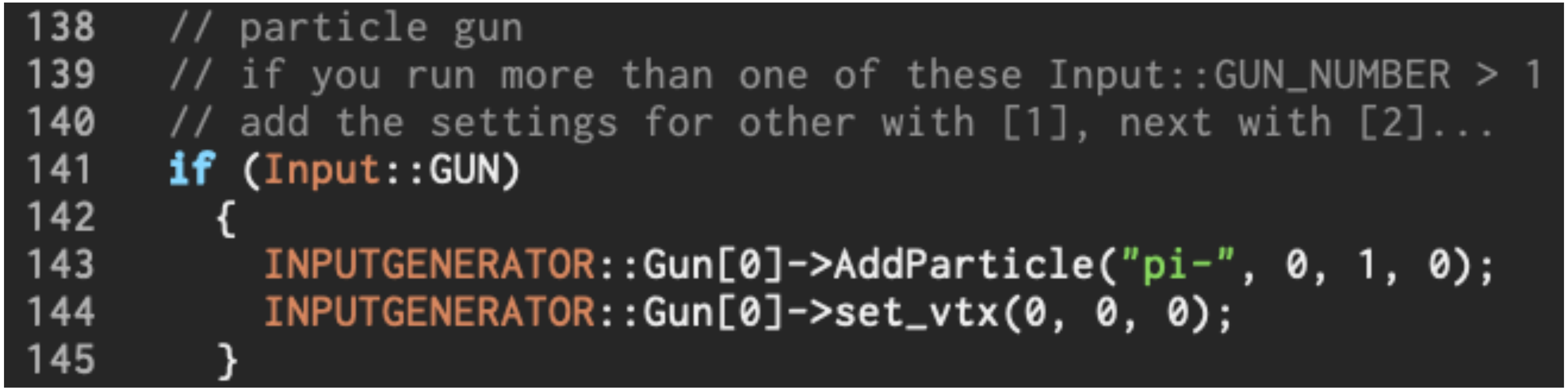
Practical example2: MC, Event generator

Event generators:

- GUN: A particle gun to shoot particles as you want



```
Input::GUN = true;
Input::GUN_NUMBER = 3; // if you need 3 of them
Input::GUN_VERBOSITY = 1;
```



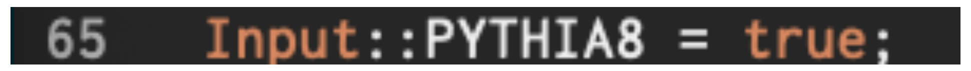
```
// particle gun
// if you run more than one of these Input::GUN_NUMBER > 1
// add the settings for other with [1], next with [2]...
if (Input::GUN)
  {
    INPUTGENERATOR::Gun[0]->AddParticle("pi-", 0, 1, 0);
    INPUTGENERATOR::Gun[0]->set_vtx(0, 0, 0);
  }
```

Configuration for GUN generator

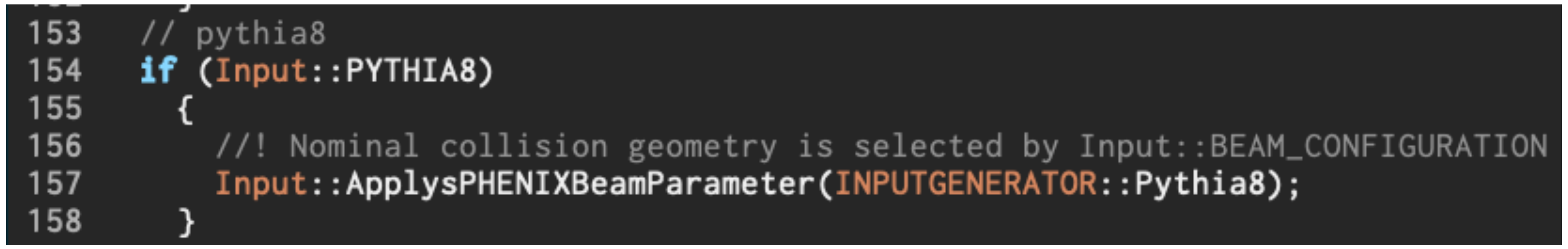
Practical example2: MC, Event generator

Event generators:

- Pythia8: General event generator

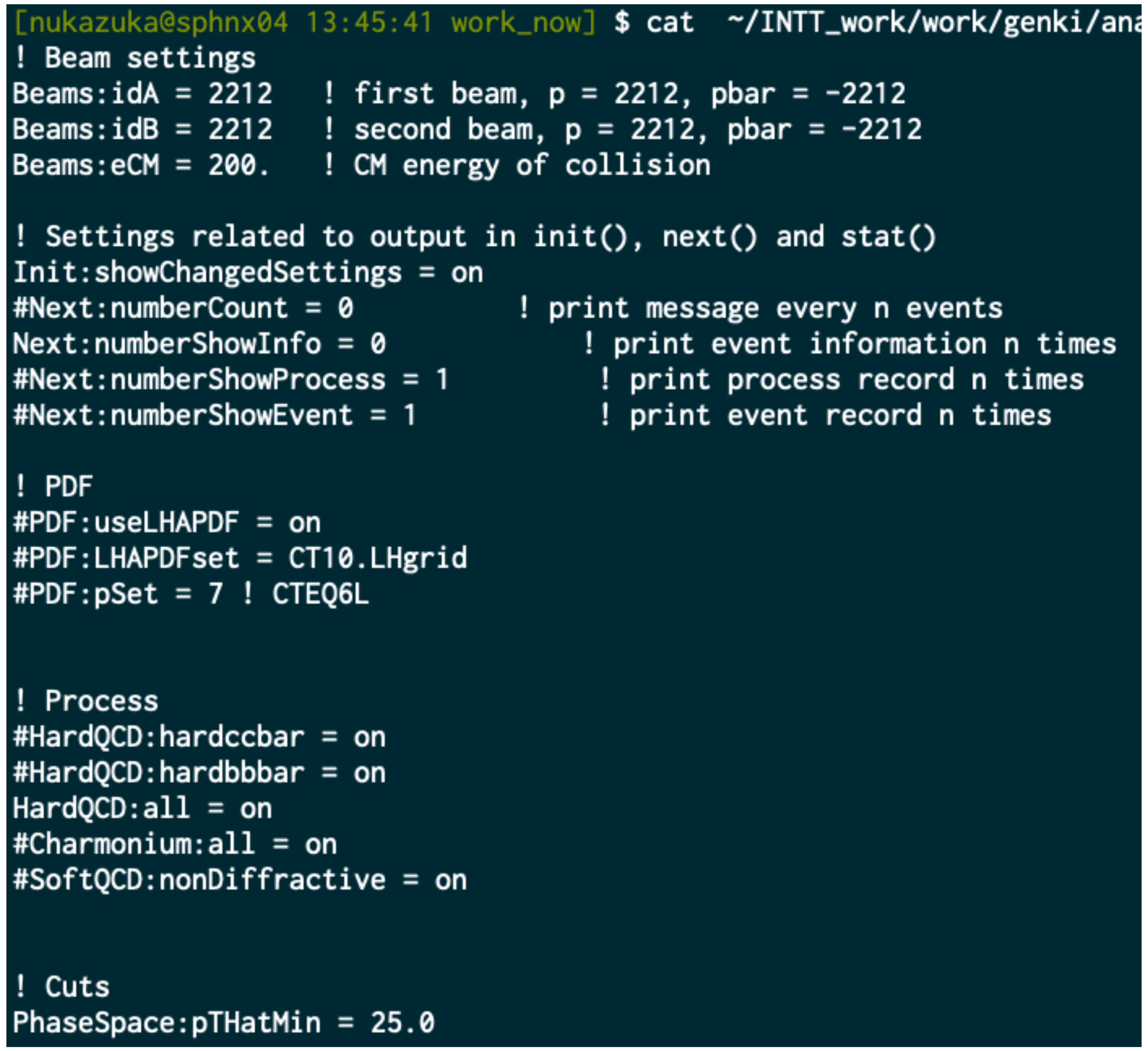


```
Input::PYTHIA8 = true;
```



```
// pythia8
if (Input::PYTHIA8)
  {
    //! Nominal collision geometry is selected by Input::BEAM_CONFIGURATION
    Input::ApplysPHENIXBeamParameter(INPUTGENERATOR::Pythia8);
  }
```

Configuration for Pythia8 generator
The real configuration is done through a text file.
You need to generate libPHPythia8.so by yourself (maybe) by compiling files in [generators/PHPythia8]() in the sPHENIX coresoftware repository.



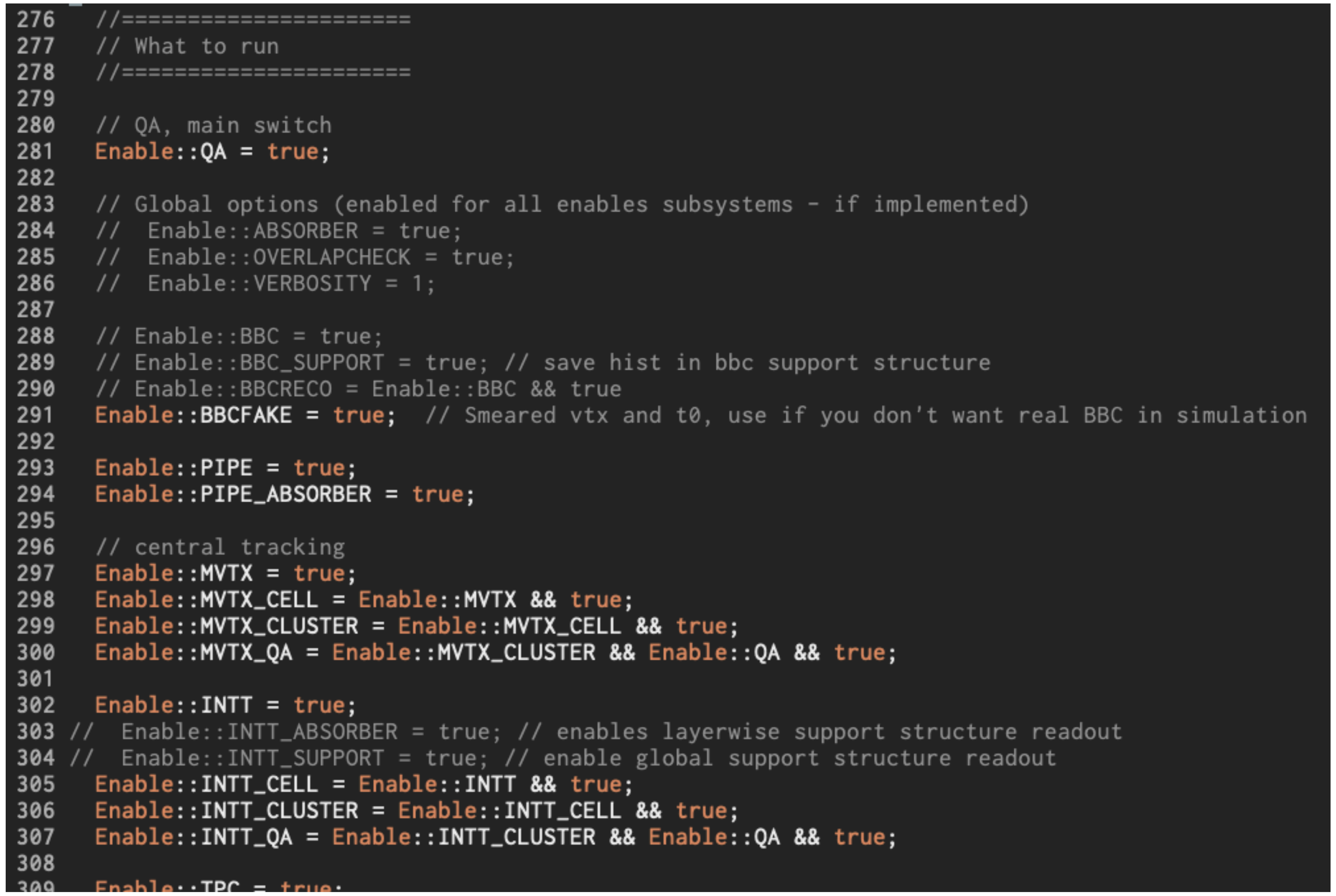
```
[nukazuka@sphnx04 13:45:41 work_now] $ cat  ~/INTT_work/work/genki/ana
! Beam settings
Beams:idA = 2212    ! first beam, p = 2212, pbar = -2212
Beams:idB = 2212    ! second beam, p = 2212, pbar = -2212
Beams:eCM = 200.    ! CM energy of collision

! Settings related to output in init(), next() and stat()
Init:showChangedSettings = on
#Next:numberCount = 0           ! print message every n events
Next:numberShowInfo = 0              ! print event information n times
#Next:numberShowProcess = 1           ! print process record n times
#Next:numberShowEvent = 1             ! print event record n times

! PDF
#PDF:useLHAPDF = on
#PDF:LHAPDFset = CT10.LHgrid
#PDF:pSet = 7 ! CTEQ6L


! Process
#HardQCD:hardccbar = on
#HardQCD:hardbbbar = on
HardQCD:all = on
#Charmonium:all = on
#SoftQCD:nonDiffractive = on


! Cuts
PhaseSpace:pTHatMin = 25.0
```

The default configuration file:
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana/*/share/calibrations/Generators/phpythia8.cfg

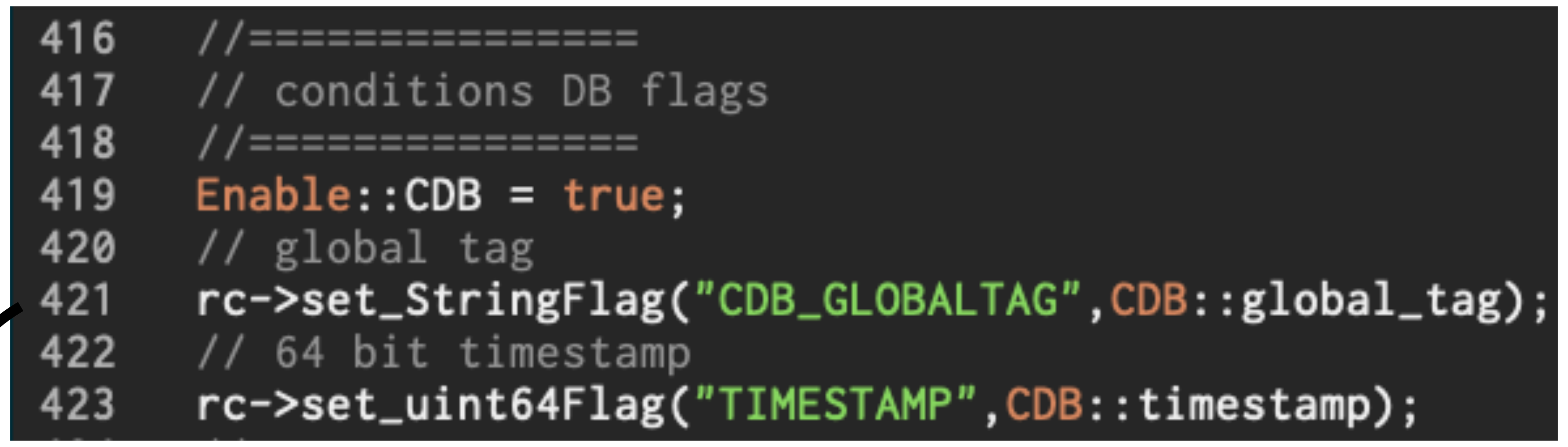
Practical example2: MC, Geometry

I’m not quite sure what is really needed for the geometry configuration. Probably,

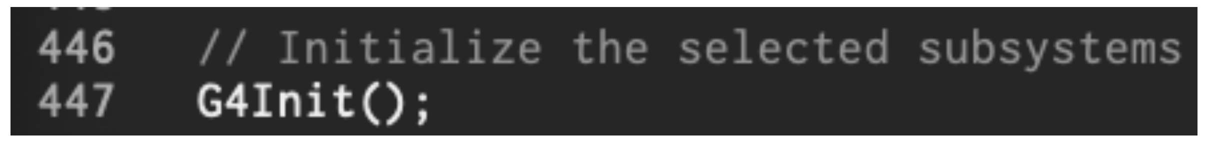
enabling detectors by assigning true to the variables in the Enable namespace (defined in multiple files, maybe files in common of the sPHENIX macros repository) is necessary.



```
  //======================
  // What to run
  //======================

  // QA, main switch
  Enable::QA = true;

  // Global options (enabled for all enables subsystems - if implemented)
  //  Enable::ABSORBER = true;
  //  Enable::OVERLAPCHECK = true;
  //  Enable::VERBOSITY = 1;

  // Enable::BBC = true;
  // Enable::BBC_SUPPORT = true; // save hist in bbc support structure
  // Enable::BBCRECO = Enable::BBC && true
  Enable::BBCFAKE = true;  // Smeared vtx and t0, use if you don't want real BBC in simulation

  Enable::PIPE = true;
  Enable::PIPE_ABSORBER = true;

  // central tracking
  Enable::MVTX = true;
  Enable::MVTX_CELL = Enable::MVTX && true;
  Enable::MVTX_CLUSTER = Enable::MVTX_CELL && true;
  Enable::MVTX_QA = Enable::MVTX_CLUSTER && Enable::QA && true;

  Enable::INTT = true;
//  Enable::INTT_ABSORBER = true; // enables layerwise support structure readout
//  Enable::INTT_SUPPORT = true; // enable global support structure readout
  Enable::INTT_CELL = Enable::INTT && true;
  Enable::INTT_CLUSTER = Enable::INTT_CELL && true;
  Enable::INTT_QA = Enable::INTT_CLUSTER && Enable::QA && true;

  Enable::TPC = true;
```



```
  //===============
  // conditions DB flags
  //===============
  Enable::CDB = true;
  // global tag
  rc->set_StringFlag("CDB_GLOBALTAG",CDB::global_tag);
  // 64 bit timestamp
  rc->set_uint64Flag("TIMESTAMP",CDB::timestamp);
```

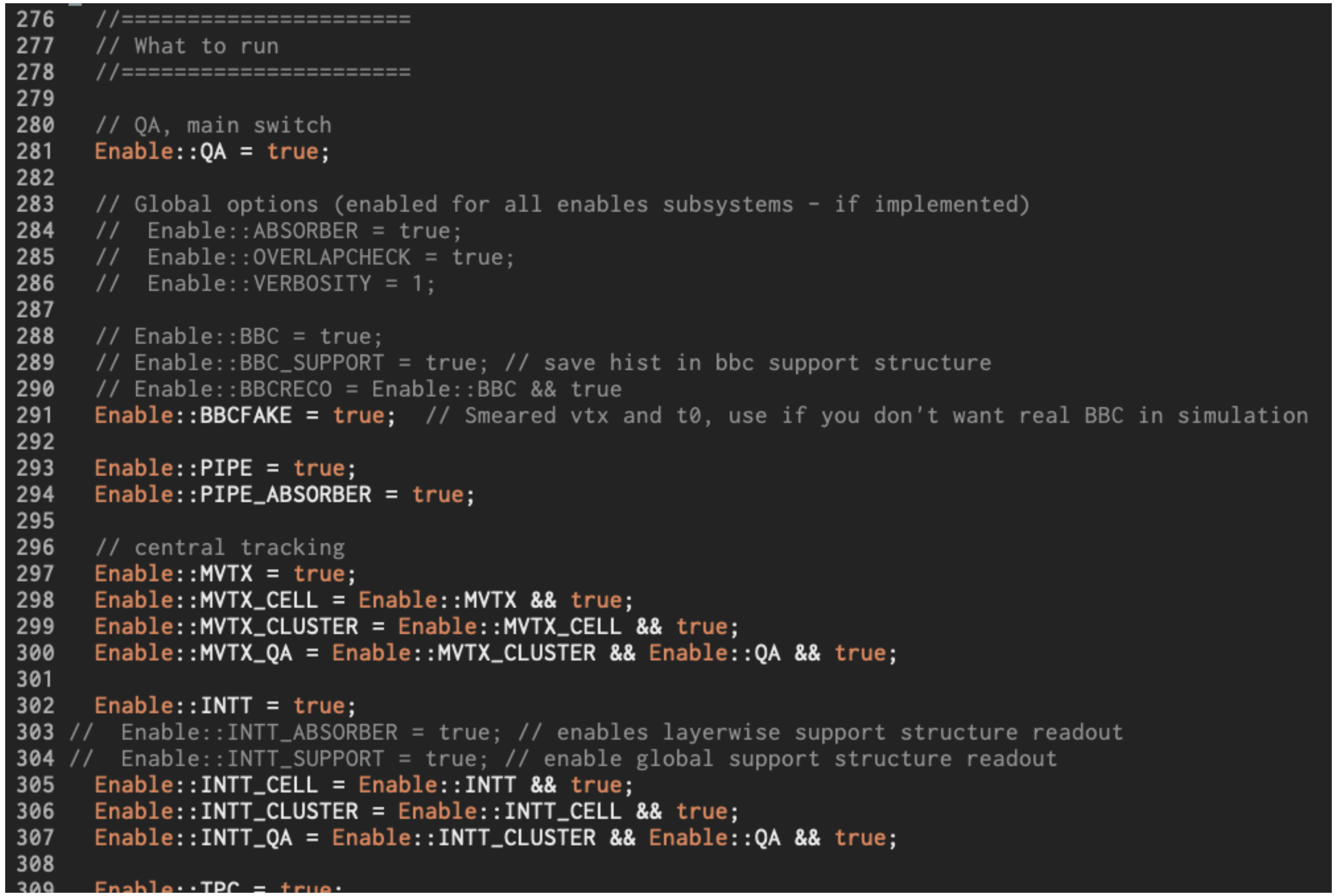


```
  // Initialize the selected subsystems
  G4Init();
```

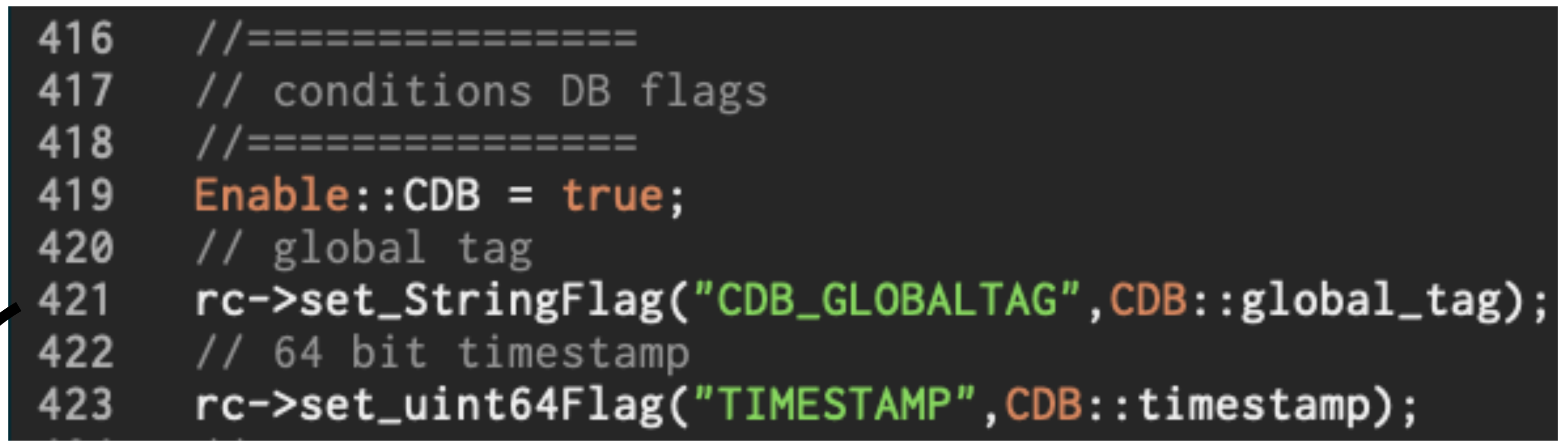
Practical example2: MC, Geometry

I’m not quite sure what is really needed for the geometry configuration. Probably,

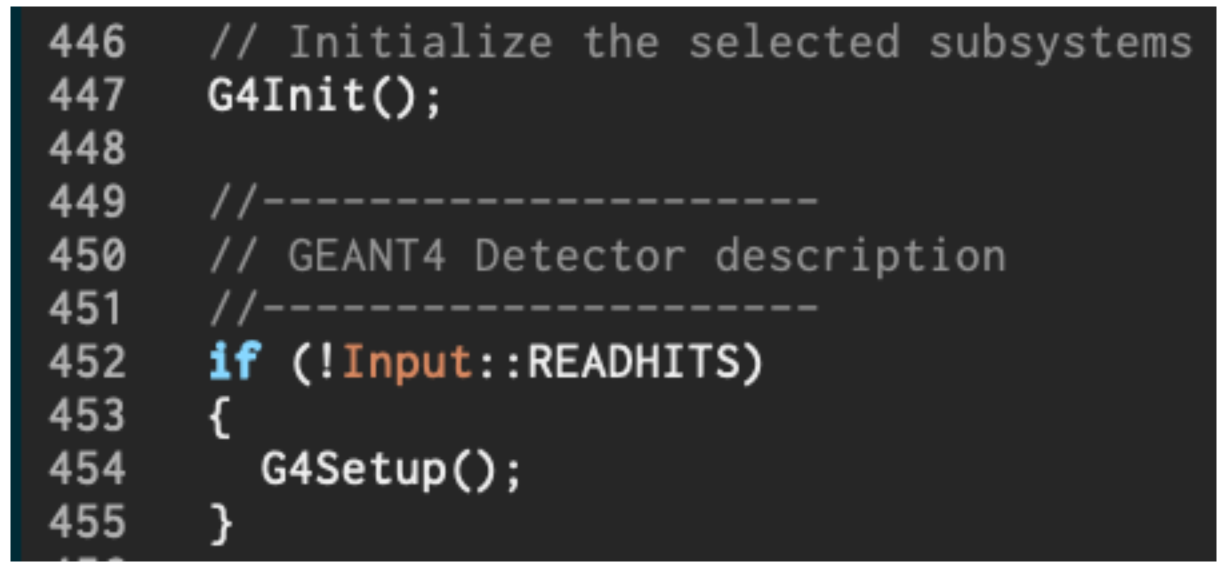
enabling detectors by assigning true to the variables in the Enable namespace (defined in multiple files, maybe files in common of the sPHENIX macros repository) is necessary.



```
  //======================
  // What to run
  //======================

  // QA, main switch
  Enable::QA = true;

  // Global options (enabled for all enables subsystems - if implemented)
  //  Enable::ABSORBER = true;
  //  Enable::OVERLAPCHECK = true;
  //  Enable::VERBOSITY = 1;

  // Enable::BBC = true;
  // Enable::BBC_SUPPORT = true; // save hist in bbc support structure
  // Enable::BBCRECO = Enable::BBC && true
  Enable::BBCFAKE = true;  // Smeared vtx and t0, use if you don't want real BBC in simulation

  Enable::PIPE = true;
  Enable::PIPE_ABSORBER = true;

  // central tracking
  Enable::MVTX = true;
  Enable::MVTX_CELL = Enable::MVTX && true;
  Enable::MVTX_CLUSTER = Enable::MVTX_CELL && true;
  Enable::MVTX_QA = Enable::MVTX_CLUSTER && Enable::QA && true;

  Enable::INTT = true;
//  Enable::INTT_ABSORBER = true; // enables layerwise support structure readout
//  Enable::INTT_SUPPORT = true; // enable global support structure readout
  Enable::INTT_CELL = Enable::INTT && true;
  Enable::INTT_CLUSTER = Enable::INTT_CELL && true;
  Enable::INTT_QA = Enable::INTT_CLUSTER && Enable::QA && true;

  Enable::TPC = true;
```



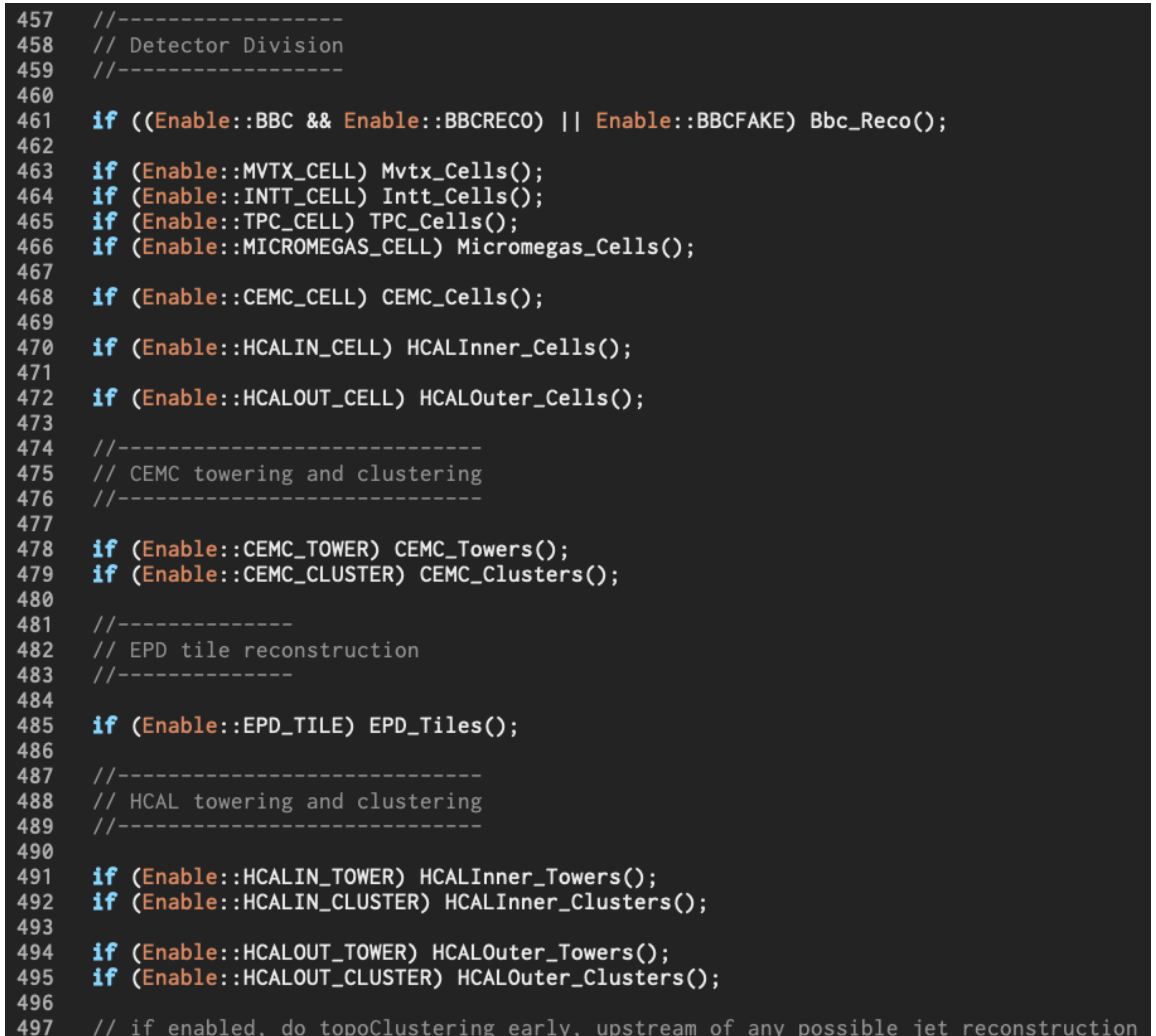
```
  //===============
  // conditions DB flags
  //===============
  Enable::CDB = true;
  // global tag
  rc->set_StringFlag("CDB_GLOBALTAG",CDB::global_tag);
  // 64 bit timestamp
  rc->set_uint64Flag("TIMESTAMP",CDB::timestamp);
```



```
  // Initialize the selected subsystems
  G4Init();

  //---------------------
  // GEANT4 Detector description
  //---------------------
  if (!Input::READHITS)
  {
    G4Setup();
  }
```

Practical example2: MC, Detector behavior

The detector behavior such as clustering, needs to be enabled.



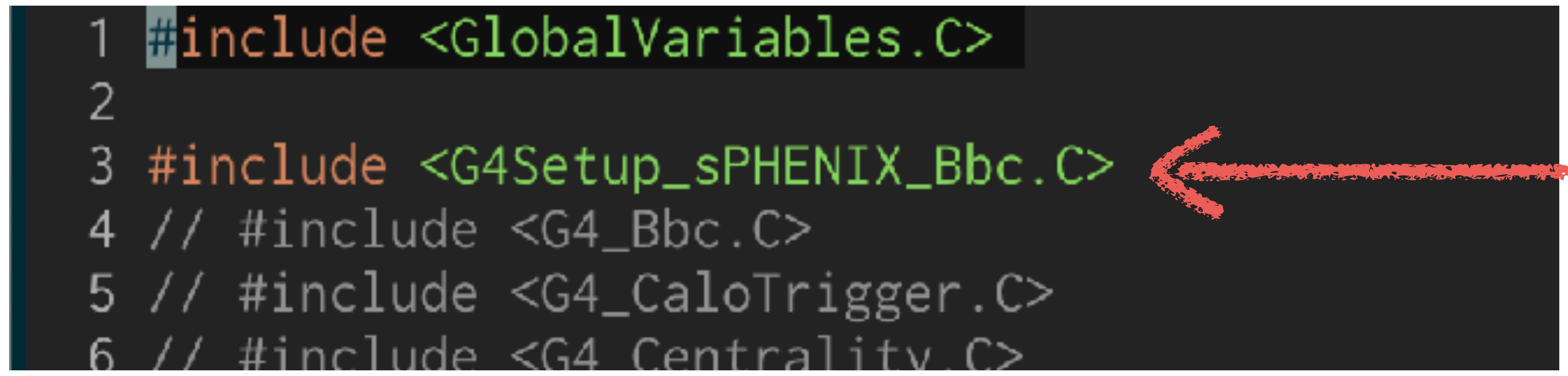
```
  //------------------
  // Detector Division
  //------------------

  if ((Enable::BBC && Enable::BBCRECO) || Enable::BBCFAKE) Bbc_Reco();

  if (Enable::MVTX_CELL) Mvtx_Cells();
  if (Enable::INTT_CELL) Intt_Cells();
  if (Enable::TPC_CELL) TPC_Cells();
  if (Enable::MICROMEGAS_CELL) Micromegas_Cells();

  if (Enable::CEMC_CELL) CEMC_Cells();

  if (Enable::HCALIN_CELL) HCALInner_Cells();

  if (Enable::HCALOUT_CELL) HCALOuter_Cells();

  //-----------------------------
  // CEMC towering and clustering
  //-----------------------------

  if (Enable::CEMC_TOWER) CEMC_Towers();
  if (Enable::CEMC_CLUSTER) CEMC_Clusters();

  //--------------
  // EPD tile reconstruction
  //--------------

  if (Enable::EPD_TILE) EPD_Tiles();

  //-----------------------------
  // HCAL towering and clustering
  //-----------------------------

  if (Enable::HCALIN_TOWER) HCALInner_Towers();
  if (Enable::HCALIN_CLUSTER) HCALInner_Clusters();

  if (Enable::HCALOUT_TOWER) HCALOuter_Towers();
  if (Enable::HCALOUT_CLUSTER) HCALOuter_Clusters();

  // if enabled, do topoClustering early, upstream of any possible jet reconstruction
```

enabling detectors by assigning true to the variables in the Enable namespace (defined in multiple files, maybe files in common of the sPHENIX macros repository) is necessary.

Practical example2: MC, Detector behavior

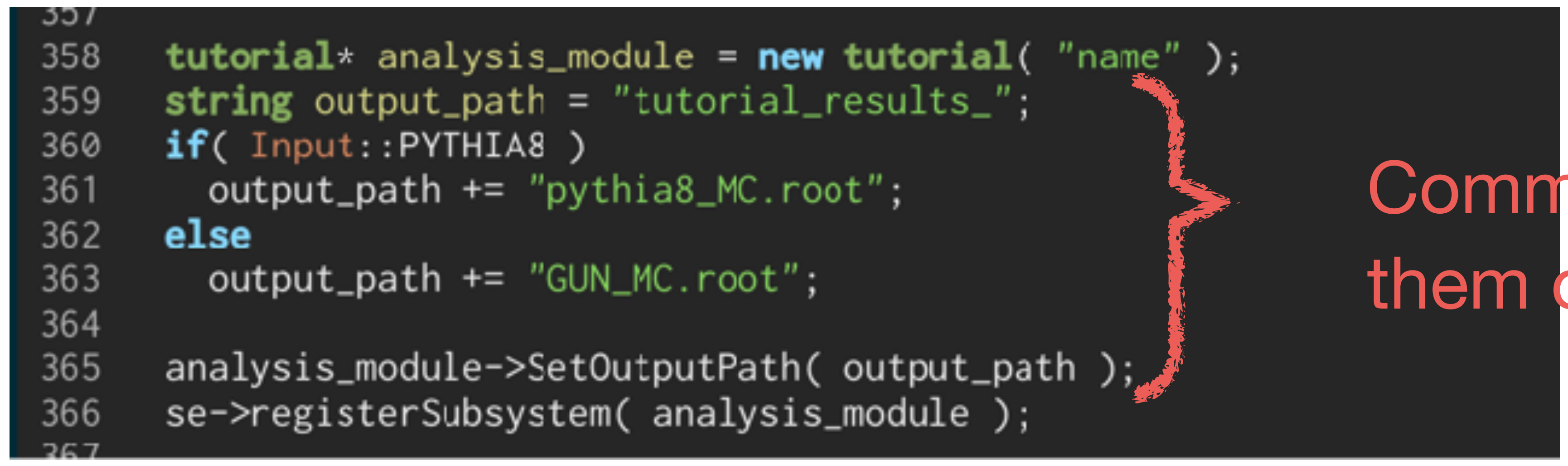
HANDS ON!

Let’s run Fun4All_minimum_3.C.



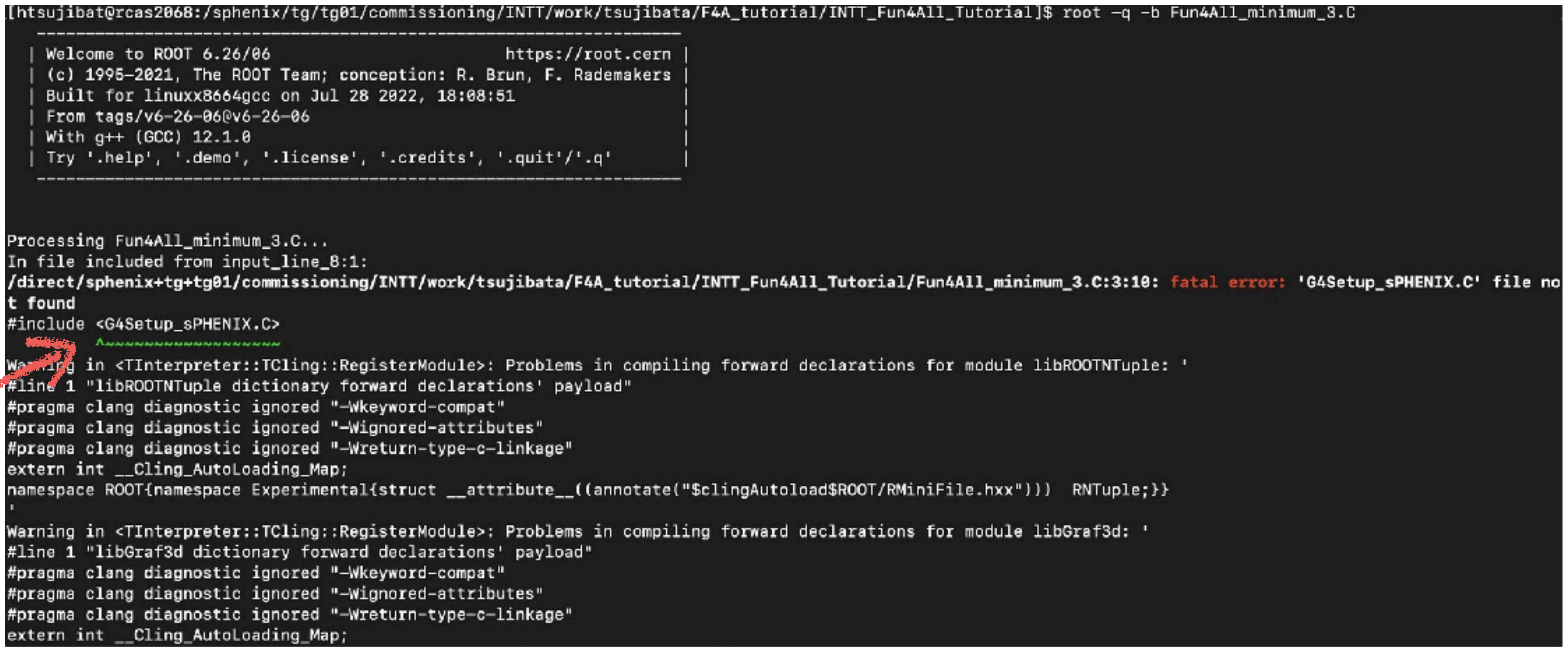
```
#include <GlobalVariables.C>

#include <G4Setup_sPHENIX_Bbc.C>
// #include <G4_Bbc.C>
// #include <G4_CaloTrigger.C>
// #include <G4_Centrality.C>
```

Please use G4Setup_sPHENIX.C.



```
  tutorial* analysis_module = new tutorial( "name" );
  string output_path = "tutorial_results_";
  if( Input::PYTHIA8 )
    output_path += "pythia8_MC.root";
  else
    output_path += "GUN_MC.root";

  analysis_module->SetOutputPath( output_path );
  se->registerSubsystem( analysis_module );
```

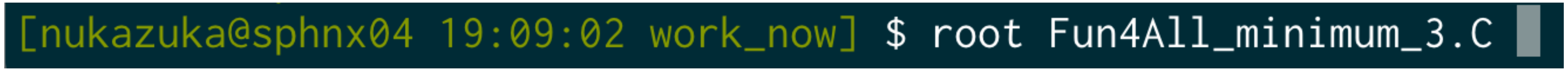
Comment them out



```
[htsujibat@rcas2068:/sphenix/tg/tg01/commissioning/INTT/work/tsujibata/F4A_tutorial/INTT_Fun4All_Tutorial]$ root -q -b Fun4All_minimum_3.C
   ------------------------------------------------------------------
  | Welcome to ROOT 6.26/06                        https://root.cern |
  | (c) 1995-2021, The ROOT Team; conception: R. Brun, F. Rademakers |
  | Built for linuxx8664gcc on Jul 28 2022, 18:08:51                 |
  | From tags/v6-26-06@v6-26-06                                      |
  | With g++ (GCC) 12.1.0                                            |
  | Try '.help', '.demo', '.license', '.credits', '.quit'/'.q'       |
   ------------------------------------------------------------------


Processing Fun4All_minimum_3.C...
In file included from input_line_8:1:
/direct/sphenix+tg+tg01/commissioning/INTT/work/tsujibata/F4A_tutorial/INTT_Fun4All_Tutorial/Fun4All_minimum_3.C:3:10: fatal error: 'G4Setup_sPHENIX.C' file no
t found
#include <G4Setup_sPHENIX.C>
         ^~~~~~~~~~~~~~~~~~~
Warning in <TInterpreter::TCling::RegisterModule>: Problems in compiling forward declarations for module libROOTNTuple: '
#line 1 "libROOTNTuple dictionary forward declarations' payload"
#pragma clang diagnostic ignored "-Wkeyword-compat"
#pragma clang diagnostic ignored "-Wignored-attributes"
#pragma clang diagnostic ignored "-Wreturn-type-c-linkage"
extern int __Cling_AutoLoading_Map;
namespace ROOT{namespace Experimental{struct __attribute__((annotate("$clingAutoload$ROOT/RMiniFile.hxx")))  RNTuple;}}
'
Warning in <TInterpreter::TCling::RegisterModule>: Problems in compiling forward declarations for module libGraf3d: '
#line 1 "libGraf3d dictionary forward declarations' payload"
#pragma clang diagnostic ignored "-Wkeyword-compat"
#pragma clang diagnostic ignored "-Wignored-attributes"
#pragma clang diagnostic ignored "-Wreturn-type-c-linkage"
extern int __Cling_AutoLoading_Map;
```

If you see such an error, modify ROOT_INCLUDE_PATH.

```
export ROOT_INCLUDE_PATH=/sphenix/tg/tg01/commissioning/INTT/repositories/macros/detectors/sPHENIX:${ROOT_INCLUDE_PATH}
```

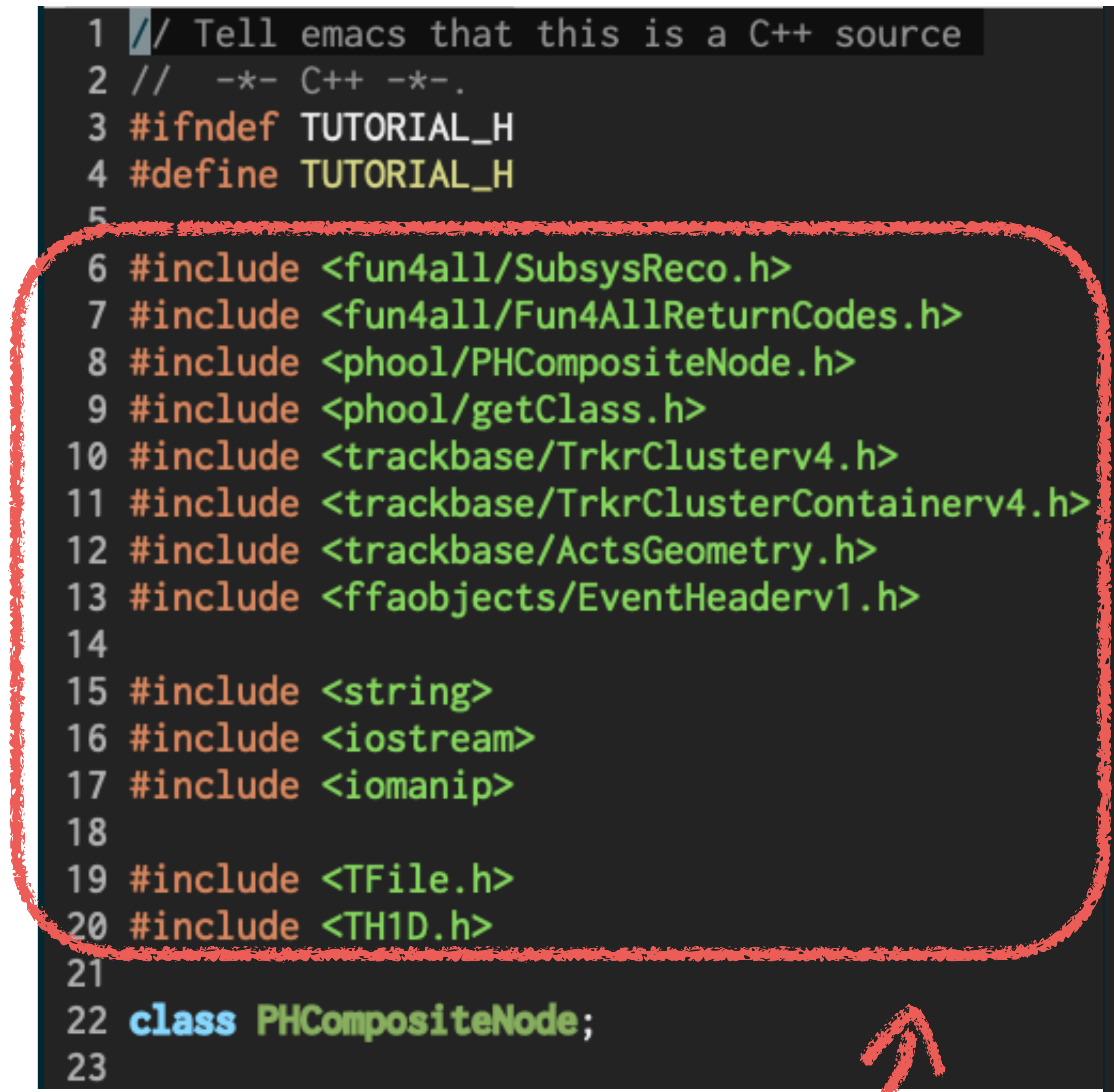


```
[nukazuka@sphnx04 19:09:02 work_now] $ root Fun4All_minimum_3.C
```

Practical example2: MC, Analysis module

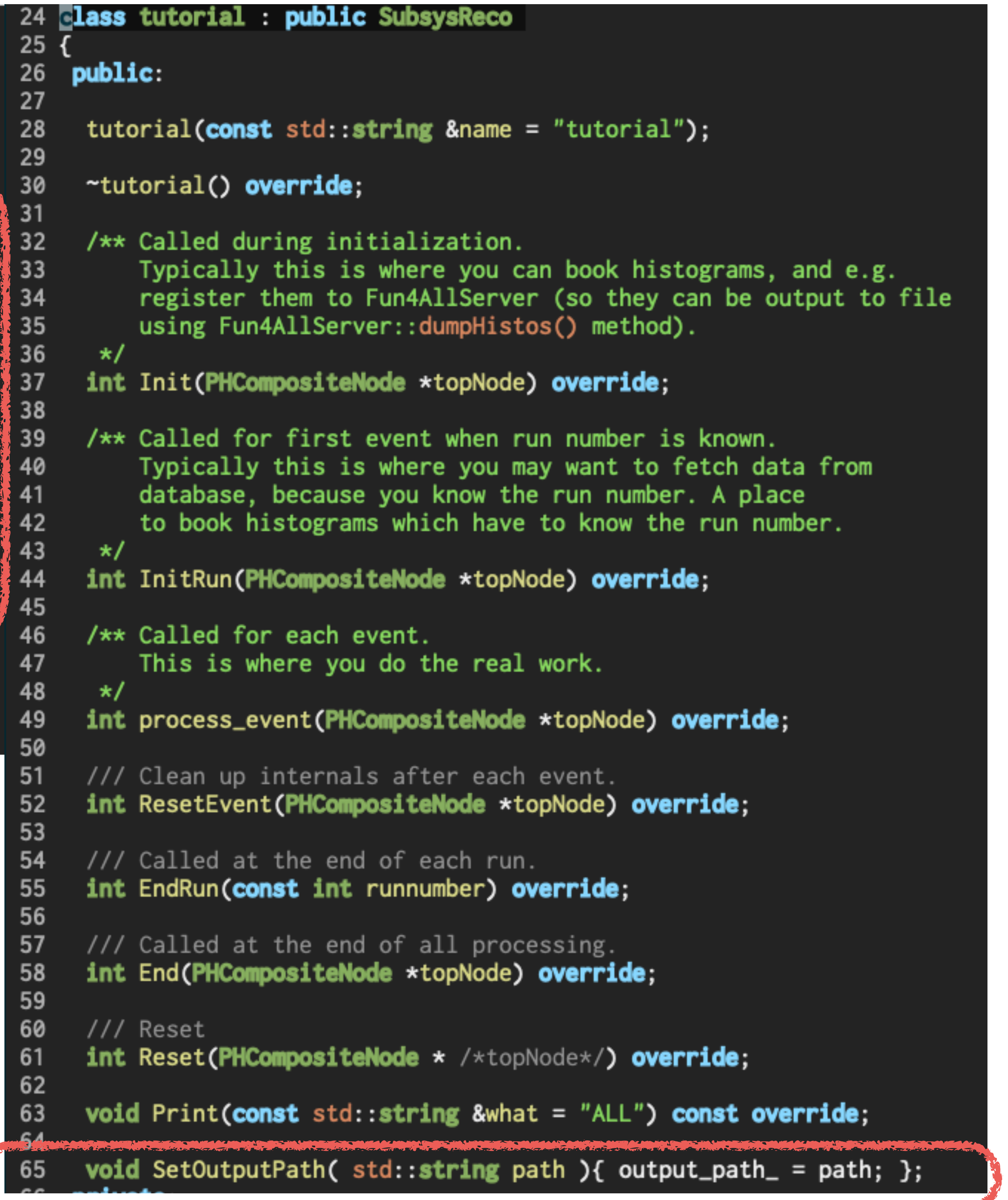
ref: sample_module_3/tutorial.h

Let's change your analysis module for MC data analysis.



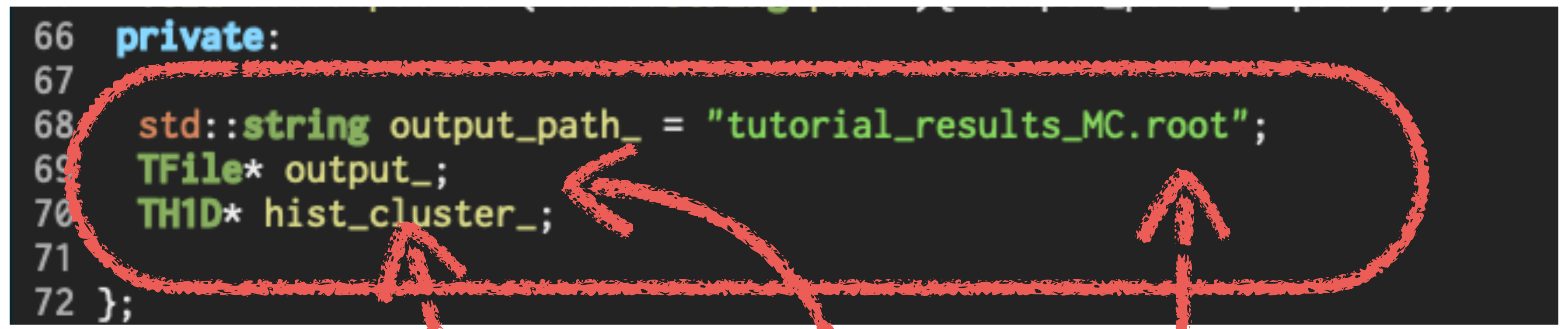
```
// Tell emacs that this is a C++ source
//  -*- C++ -*-.
#ifndef TUTORIAL_H
#define TUTORIAL_H

#include <fun4all/SubsysReco.h>
#include <fun4all/Fun4AllReturnCodes.h>
#include <phool/PHCompositeNode.h>
#include <phool/getClass.h>
#include <trackbase/TrkrClusterv4.h>
#include <trackbase/TrkrClusterContainerv4.h>
#include <trackbase/ActsGeometry.h>
#include <ffaobjects/EventHeaderv1.h>

#include <string>
#include <iostream>
#include <iomanip>

#include <TFile.h>
#include <TH1D.h>

class PHCompositeNode;

```

Some headers were added.



```
class tutorial : public SubsysReco
{
 public:

  tutorial(const std::string &name = "tutorial");

  ~tutorial() override;

  /** Called during initialization.
      Typically this is where you can book histograms, and e.g.
      register them to Fun4AllServer (so they can be output to file
      using Fun4AllServer::dumpHistos() method).
   */
  int Init(PHCompositeNode *topNode) override;

  /** Called for first event when run number is known.
      Typically this is where you may want to fetch data from
      database, because you know the run number. A place
      to book histograms which have to know the run number.
   */
  int InitRun(PHCompositeNode *topNode) override;

  /** Called for each event.
      This is where you do the real work.
   */
  int process_event(PHCompositeNode *topNode) override;

  /// Clean up internals after each event.
  int ResetEvent(PHCompositeNode *topNode) override;

  /// Called at the end of each run.
  int EndRun(const int runnumber) override;

  /// Called at the end of all processing.
  int End(PHCompositeNode *topNode) override;

  /// Reset
  int Reset(PHCompositeNode * /*topNode*/) override;

  void Print(const std::string &what = "ALL") const override;

  void SetOutputPath( std::string path ){ output_path_ = path; };
```

A function to set the output path was added.



```
 private:

  std::string output_path_ = "tutorial_results_MC.root";
  TFile* output_;
  TH1D* hist_cluster_;

};
```

The output path.

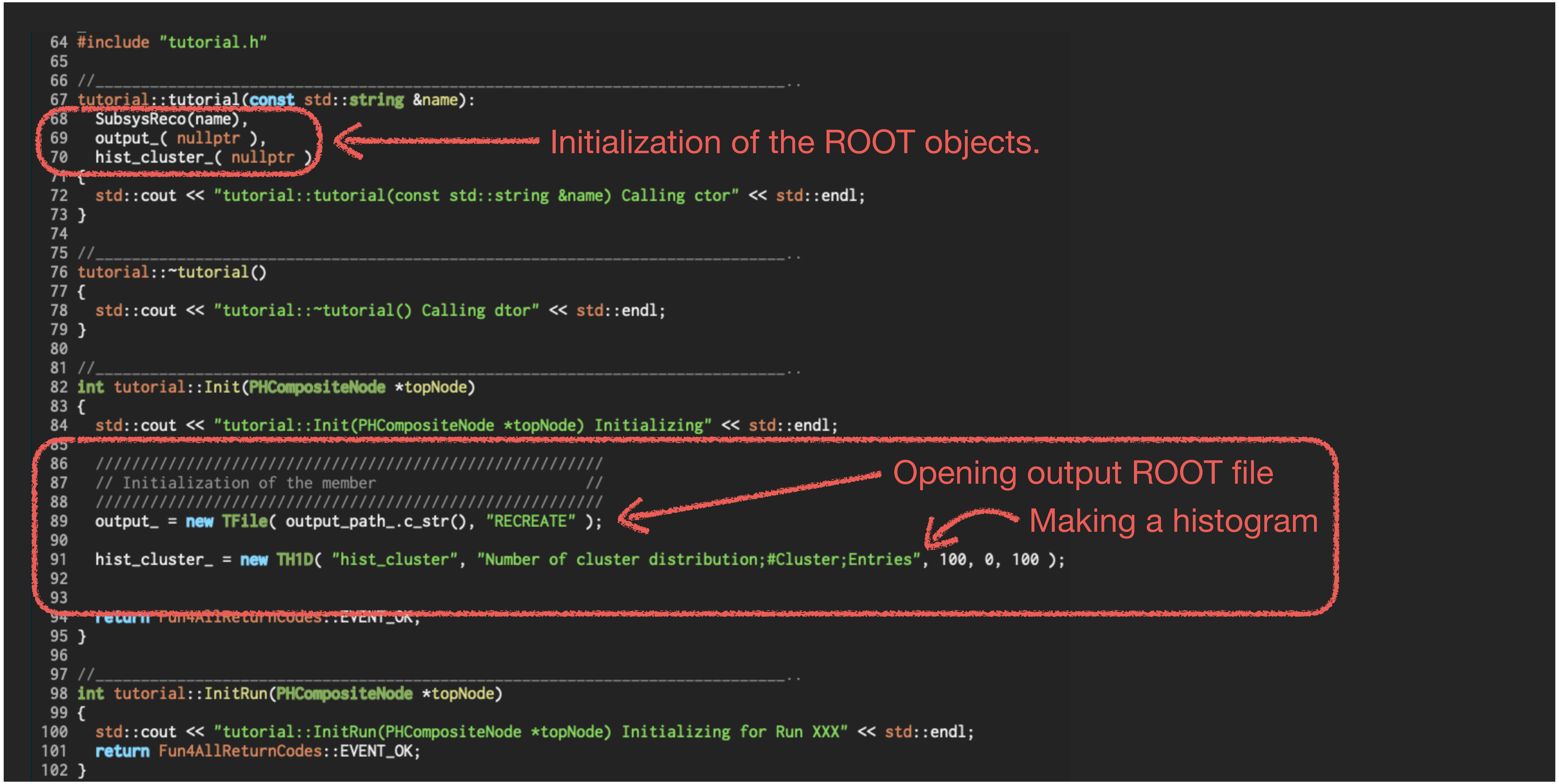
TFile* object for output.

TH1D* object to contain the analysis results

Practical example2: MC, Analysis module

ref: sample_module_3/tutorial.cc

Let’s change your analysis module for MC data analysis.



```
#include "tutorial.h"

//____________________________________________________________________________..
tutorial::tutorial(const std::string &name):
  SubsysReco(name),
  output_( nullptr ),
  hist_cluster_( nullptr )
{
  std::cout << "tutorial::tutorial(const std::string &name) Calling ctor" << std::endl;
}

//____________________________________________________________________________..
tutorial::~tutorial()
{
  std::cout << "tutorial::~tutorial() Calling dtor" << std::endl;
}

//____________________________________________________________________________..
int tutorial::Init(PHCompositeNode *topNode)
{
  std::cout << "tutorial::Init(PHCompositeNode *topNode) Initializing" << std::endl;

  //////////////////////////////////////////////////////////
  // Initialization of the member                         //
  //////////////////////////////////////////////////////////
  output_ = new TFile( output_path_.c_str(), "RECREATE" );

  hist_cluster_ = new TH1D( "hist_cluster", "Number of cluster distribution;#Cluster;Entries", 100, 0, 100 );


  return Fun4AllReturnCodes::EVENT_OK;
}

//____________________________________________________________________________..
int tutorial::InitRun(PHCompositeNode *topNode)
{
  std::cout << "tutorial::InitRun(PHCompositeNode *topNode) Initializing for Run XXX" << std::endl;
  return Fun4AllReturnCodes::EVENT_OK;
}
```

Initialization of the ROOT objects.

Opening output ROOT file

Making a histogram

Practical example2: MC, Analysis module

ref: sample_module_3/tutorial.cc

Let's change your analysis module for MC data analysis.

```
int tutorial::process_event(PHCompositeNode *topNode)
{
  std::cout << "tutorial::process_event(PHCompositeNode *topNode) Processing Event" << std::endl;

  /////////////////////////////////////////////////////////////////////////////////////////
  // Getting Nodes                                                                       //
  /////////////////////////////////////////////////////////////////////////////////////////
  // TRKR_CLUSTER node: Information of TrkrCluster
  auto *node_cluster_map =
    findNode::getClass<TrkrClusterContainerv4>(topNode, "TRKR_CLUSTER");

  if (!node_cluster_map)
    {
      std::cerr << PHWHERE << "TrkrClusterContainer node is missing." << std::endl;
      return Fun4AllReturnCodes::ABORTEVENT;
    }

  // ActsGeometry node: for the global coordinate
  ActsGeometry *node_acts = findNode::getClass<ActsGeometry>(topNode, "ActsGeometry");
  if ( !node_acts )
    {
      std::cout << PHWHERE << "No ActsGeometry on node tree. Bailing." << std::endl;
      return Fun4AllReturnCodes::ABORTEVENT;
    }


```

TRKR_CLUSTER node is obtained to access TrkrCluster.

If TRKR_CLUSTER node is not found, nothing is done.

The same steps but for ActsGeometry node. It's necessary (?) to convert TrkrCluster coordinate (local coordinate in the detector) to the global coordinate (sPHENIX lab frame)

Practical example2: MC, Analysis module

ref: sample_module_3/tutorial.cc

Let’s change your analysis module for MC data analysis.

```
std::vector < TrkrCluster* > clusters;
for (unsigned int inttlayer = 0; inttlayer < 4; inttlayer++)
  {
    //      cout << " INTT layer " << inttlayer << endl;
    //      int layer= ( inttlayer < 2 ? 0 : 1 );

    // loop over all hits
    for (const auto &hitsetkey : node_cluster_map->getHitSetKeys(TrkrDefs::TrkrId::inttId, inttlayer + 3) )
      {

        // type: std::pair<ConstIterator, ConstIterator> ConstRange
        // here, MMap::const_iterator ConstIterator;
        auto range = node_cluster_map->getClusters(hitsetkey);

        // loop over iterators of this cluster
        for (auto clusIter = range.first; clusIter != range.second; ++clusIter)
          {
            const auto cluskey = clusIter->first;
            const auto cluster = clusIter->second;
            clusters.push_back( cluster );

            const auto globalPos = node_acts->getGlobalPosition(cluskey, cluster);

            // int ladder_z   = InttDefs::getLadderZId(cluskey);
            // int ladder_phi = InttDefs::getLadderPhiId(cluskey);
            int size        = cluster->getSize();
```

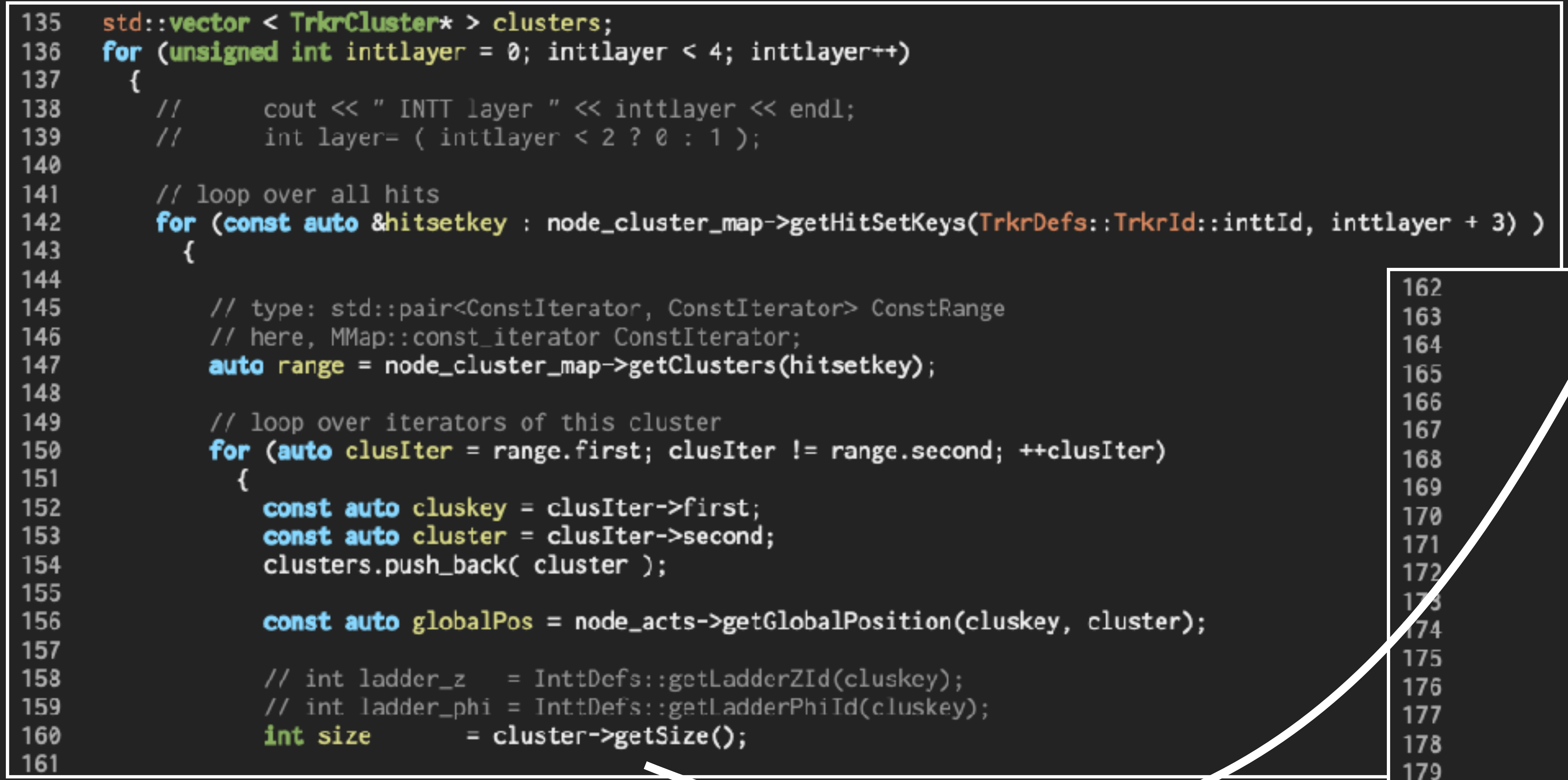
```
            //              if( nCluster < 5 )
            if (Verbosity() > 5)
              {
                std::cout
                  // << "xyz("
                  //    << std::setprecision(4) << std::setw(8) << globalPos.x() << ", "
                  //    << std::setprecision(4) << std::setw(8) << globalPos.y() << ", "
                  //    << std::setprecision(4) << std::setw(8) << globalPos.z()
                  //    << ") \t"
                     << "xyz("
                     << std::setprecision(4) << std::setw(8) << cluster->getPosition( 0 ) << ", "
                     << std::setprecision(4) << std::setw(8) << cluster->getPosition( 1 ) << ", "
                     << std::setprecision(4) << std::setw(8) << cluster->getPosition( 2 ) << ") "
                     << "local xy("
                     << std::setprecision(4) << std::setw(8) << cluster->getLocalX() << ", "
                     << std::setprecision(4) << std::setw(8) << cluster->getLocalY() << ")\t "

                     << cluster->getAdc() << " "
                     << size << " "
                     << inttlayer << " "
                     // << ladder_z << " "
                     // << ladder_phi
                     << std::endl;
              }

            cluster->setPosition(0,  globalPos.x() );
            cluster->setPosition(1,  globalPos.y() );
            cluster->setPosition(2,  globalPos.z() );
          }
      }
  }

std::cout << clusters.size() << " clusters in this event" << std::endl;
hist_cluster_->Fill( clusters.size() );

return Fun4AllReturnCodes::EVENT_OK;
}
```

```
for( loop over INTT layers ){
  for( loop over hit set ){
    for( loop over iterator of the hit set ){
      /* processes cluster by cluster */
      /* filling the histogram */
    }
  }
}
```

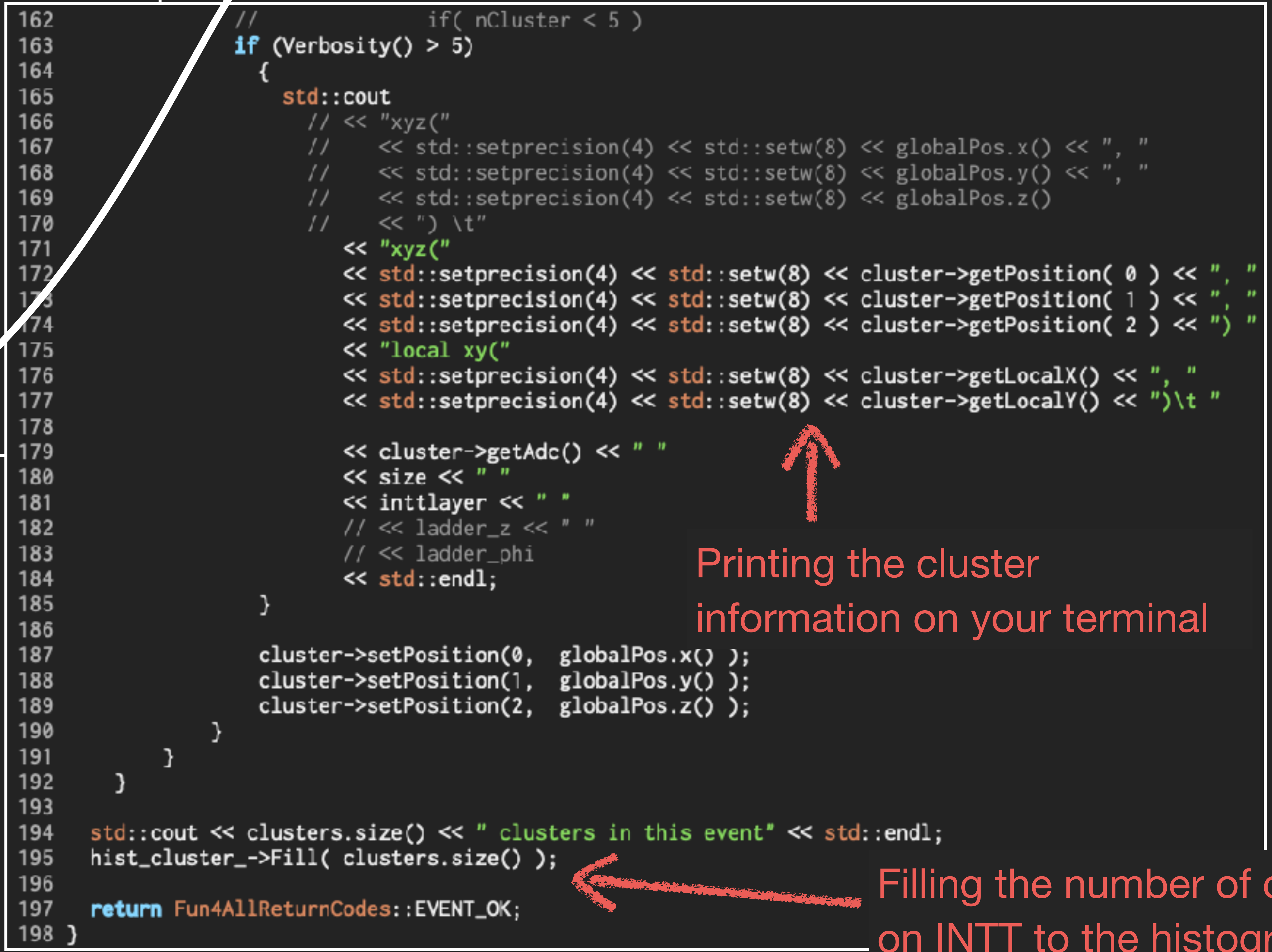
Practical example2: MC, Analysis module

ref: sample_module_3/tutorial.cc

Let’s change your analysis module for MC data analysis.



```
std::vector < TrkrCluster* > clusters;
for (unsigned int inttlayer = 0; inttlayer < 4; inttlayer++)
 {
  //      cout << " INTT layer " << inttlayer << endl;
  //      int layer= ( inttlayer < 2 ? 0 : 1 );

  // loop over all hits
  for (const auto &hitsetkey : node_cluster_map->getHitSetKeys(TrkrDefs::TrkrId::inttId, inttlayer + 3) )
   {

    // type: std::pair<ConstIterator, ConstIterator> ConstRange
    // here, MMap::const_iterator ConstIterator;
    auto range = node_cluster_map->getClusters(hitsetkey);

    // loop over iterators of this cluster
    for (auto clusIter = range.first; clusIter != range.second; ++clusIter)
     {
      const auto cluskey = clusIter->first;
      const auto cluster = clusIter->second;
      clusters.push_back( cluster );

      const auto globalPos = node_acts->getGlobalPosition(cluskey, cluster);

      // int ladder_z   = InttDefs::getLadderZId(cluskey);
      // int ladder_phi = InttDefs::getLadderPhiId(cluskey);
      int size       = cluster->getSize();

```



```
      //              if( nCluster < 5 )
      if (Verbosity() > 5)
       {
        std::cout
         // << "xyz("
         //    << std::setprecision(4) << std::setw(8) << globalPos.x() << ", "
         //    << std::setprecision(4) << std::setw(8) << globalPos.y() << ", "
         //    << std::setprecision(4) << std::setw(8) << globalPos.z()
         //    << ") \t"
           << "xyz("
           << std::setprecision(4) << std::setw(8) << cluster->getPosition( 0 ) << ", "
           << std::setprecision(4) << std::setw(8) << cluster->getPosition( 1 ) << ", "
           << std::setprecision(4) << std::setw(8) << cluster->getPosition( 2 ) << ") "
           << "local xy("
           << std::setprecision(4) << std::setw(8) << cluster->getLocalX() << ", "
           << std::setprecision(4) << std::setw(8) << cluster->getLocalY() << ")\t "

           << cluster->getAdc() << " "
           << size << " "
           << inttlayer << " "
           // << ladder_z << " "
           // << ladder_phi
           << std::endl;
       }

       cluster->setPosition(0,  globalPos.x() );
       cluster->setPosition(1,  globalPos.y() );
       cluster->setPosition(2,  globalPos.z() );
     }
   }
 }

 std::cout << clusters.size() << " clusters in this event" << std::endl;
 hist_cluster_->Fill( clusters.size() );

 return Fun4AllReturnCodes::EVENT_OK;
}
```

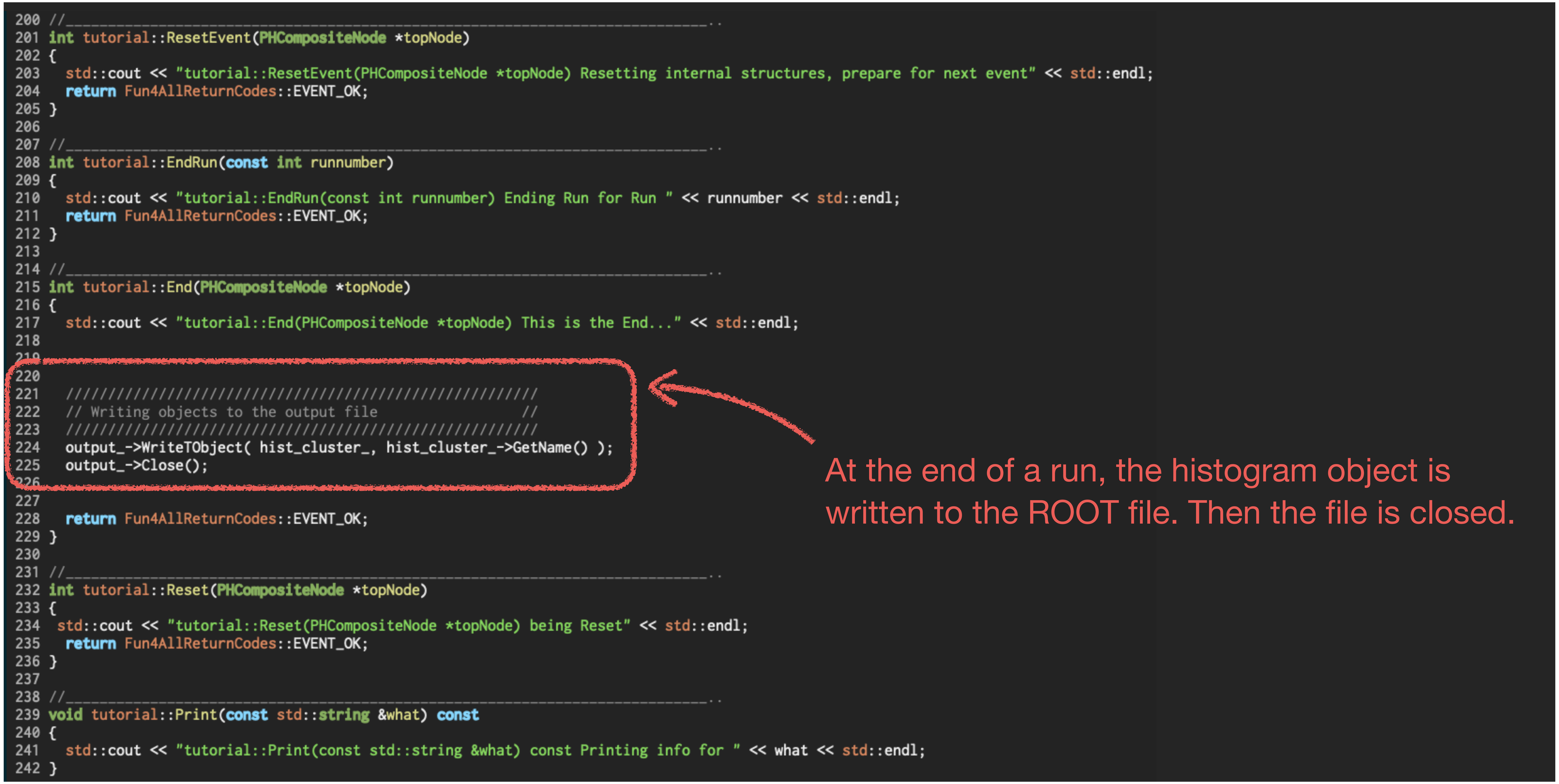
Printing the cluster information on your terminal

Filling the number of clusters on INTT to the histogram

Practical example2: MC, Analysis module

ref: sample_module_3/tutorial.cc

Let’s change your analysis module for MC data analysis.



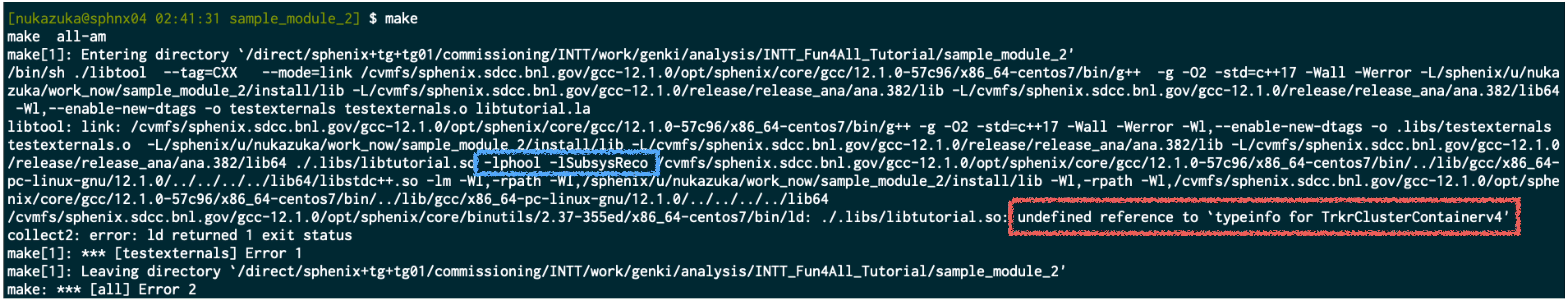
```cpp
//____________________________________________________________________________..
int tutorial::ResetEvent(PHCompositeNode *topNode)
{
  std::cout << "tutorial::ResetEvent(PHCompositeNode *topNode) Resetting internal structures, prepare for next event" << std::endl;
  return Fun4AllReturnCodes::EVENT_OK;
}

//____________________________________________________________________________..
int tutorial::EndRun(const int runnumber)
{
  std::cout << "tutorial::EndRun(const int runnumber) Ending Run for Run " << runnumber << std::endl;
  return Fun4AllReturnCodes::EVENT_OK;
}

//____________________________________________________________________________..
int tutorial::End(PHCompositeNode *topNode)
{
  std::cout << "tutorial::End(PHCompositeNode *topNode) This is the End..." << std::endl;



  ////////////////////////////////////////////////////////////////
  // Writing objects to the output file                         //
  ////////////////////////////////////////////////////////////////
  output_->WriteTObject( hist_cluster_, hist_cluster_->GetName() );
  output_->Close();


  return Fun4AllReturnCodes::EVENT_OK;
}

//____________________________________________________________________________..
int tutorial::Reset(PHCompositeNode *topNode)
{
 std::cout << "tutorial::Reset(PHCompositeNode *topNode) being Reset" << std::endl;
  return Fun4AllReturnCodes::EVENT_OK;
}

//____________________________________________________________________________..
void tutorial::Print(const std::string &what) const
{
  std::cout << "tutorial::Print(const std::string &what) const Printing info for " << what << std::endl;
}
```

At the end of a run, the histogram object is written to the ROOT file. Then the file is closed.

Compiling your analysis module

ref: sample_module_2/tutorial.h, .cc

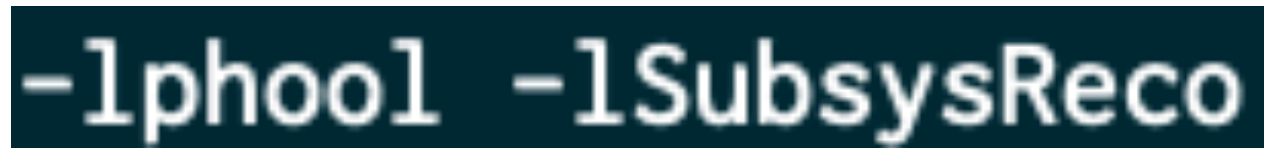
If you compile your analysis module by `$ make`, you may see the following error:



```
[nukazuka@sphnx04 02:41:31 sample_module_2] $ make
make  all-am
make[1]: Entering directory `/direct/sphenix+tg+tg01/commissioning/INTT/work/genki/analysis/INTT_Fun4All_Tutorial/sample_module_2'
/bin/sh ./libtool  --tag=CXX   --mode=link /cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/opt/sphenix/core/gcc/12.1.0-57c96/x86_64-centos7/bin/g++  -g -O2 -std=c++17 -Wall -Werror -L/sphenix/u/nuka
zuka/work_now/sample_module_2/install/lib -L/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/lib -L/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/lib64
 -Wl,--enable-new-dtags -o testexternals testexternals.o libtutorial.la
libtool: link: /cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/opt/sphenix/core/gcc/12.1.0-57c96/x86_64-centos7/bin/g++ -g -O2 -std=c++17 -Wall -Werror -Wl,--enable-new-dtags -o .libs/testexternals
testexternals.o  -L/sphenix/u/nukazuka/work_now/sample_module_2/install/lib -L/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/release/release_ana/ana.382/lib -L/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0
/release/release_ana/ana.382/lib64 ./.libs/libtutorial.so -lphool -lSubsysReco /cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/opt/sphenix/core/gcc/12.1.0-57c96/x86_64-centos7/bin/../lib/gcc/x86_64-
pc-linux-gnu/12.1.0/../../../../lib64/libstdc++.so -lm -Wl,-rpath -Wl,/sphenix/u/nukazuka/work_now/sample_module_2/install/lib -Wl,-rpath -Wl,/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/opt/sphe
nix/core/gcc/12.1.0-57c96/x86_64-centos7/bin/../lib/gcc/x86_64-pc-linux-gnu/12.1.0/../../../../lib64
/cvmfs/sphenix.sdcc.bnl.gov/gcc-12.1.0/opt/sphenix/core/binutils/2.37-355ed/x86_64-centos7/bin/ld: ./.libs/libtutorial.so: undefined reference to `typeinfo for TrkrClusterContainerv4'
collect2: error: ld returned 1 exit status
make[1]: *** [testexternals] Error 1
make[1]: Leaving directory `/direct/sphenix+tg+tg01/commissioning/INTT/work/genki/analysis/INTT_Fun4All_Tutorial/sample_module_2'
make: *** [all] Error 2
```

```
./.libs/libtutorial.so: undefined reference to `typeinfo for TrkrClusterContainerv4'
```

It means that you refer to TrkrClusterContainerv4 in your code, but the compiler is not informed of the actual implementation of the type (it's a class in this case). This is because the analysis module generated by the sPHENIX perl script only uses libphool.so and libSubsysReco.so libraries. You need to add something else by yourself.

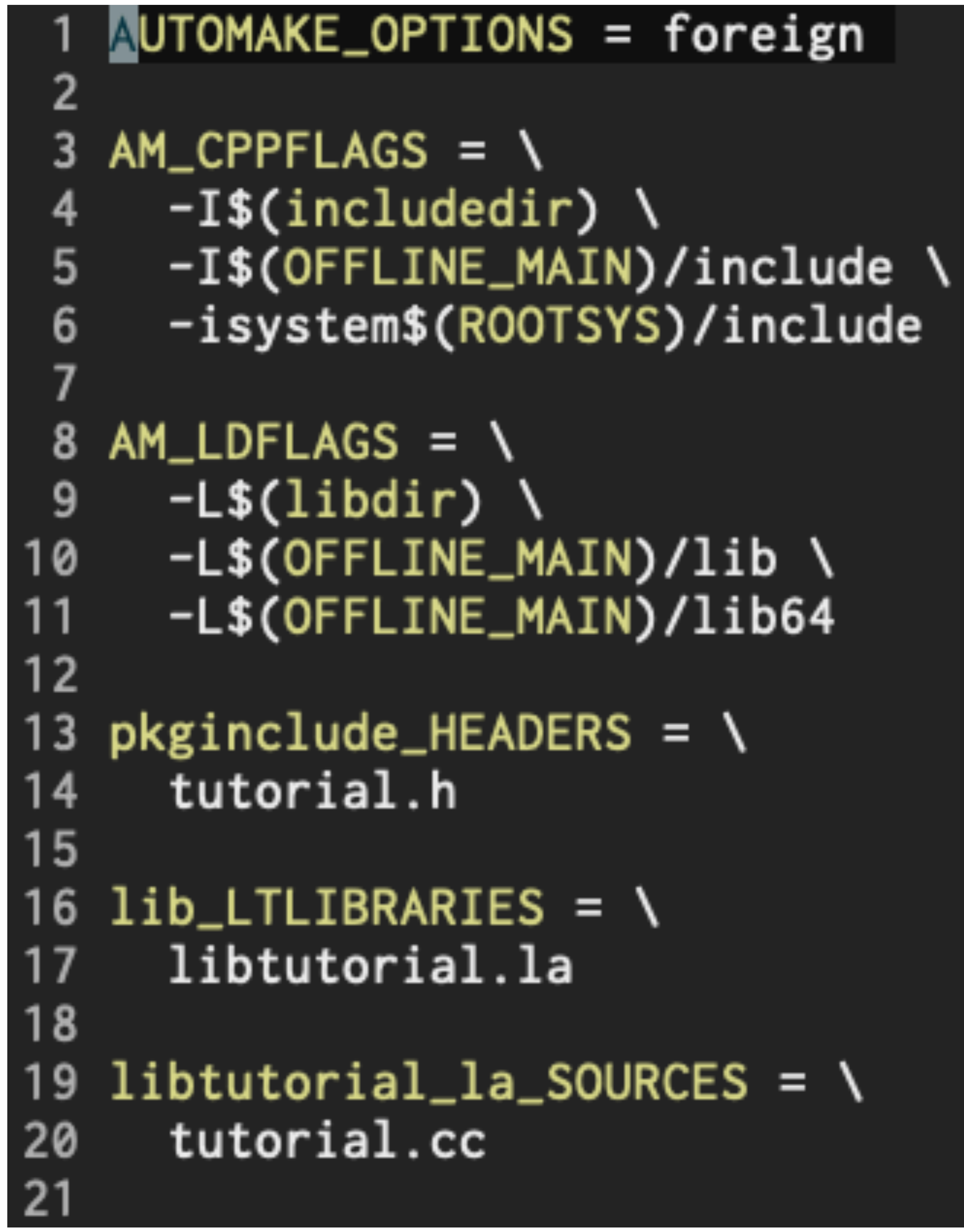


```
-lphool -lSubsysReco
```

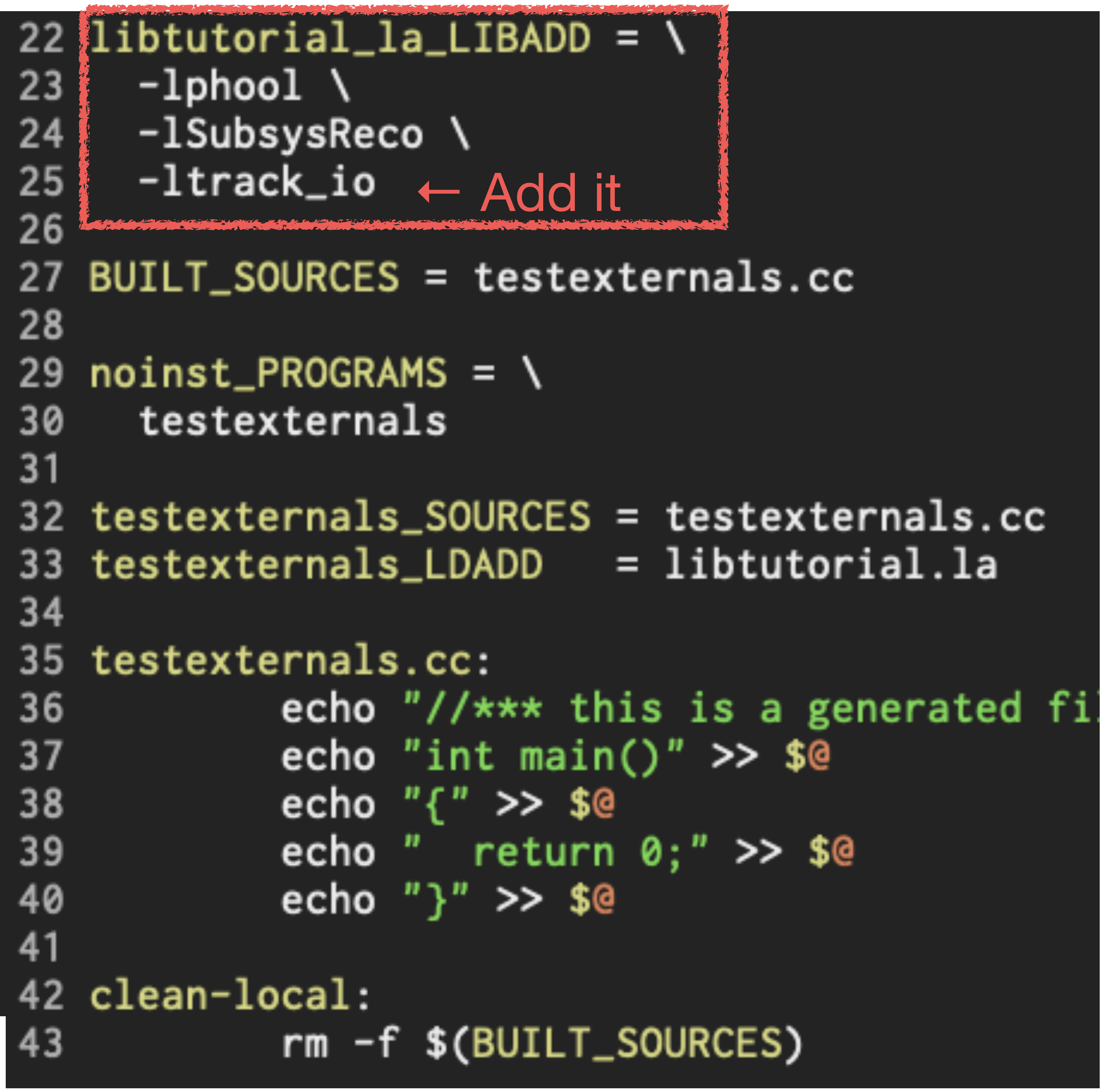
Compiling your analysis module: Makefile.am

ref: sample_module_2/tutorial.h, .cc

autogen.sh uses Makefile.am to make a Makefile that matches your environment.



```
AUTOMAKE_OPTIONS = foreign

AM_CPPFLAGS = \
  -I$(includedir) \
  -I$(OFFLINE_MAIN)/include \
  -isystem$(ROOTSYS)/include

AM_LDFLAGS = \
  -L$(libdir) \
  -L$(OFFLINE_MAIN)/lib \
  -L$(OFFLINE_MAIN)/lib64

pkginclude_HEADERS = \
  tutorial.h

lib_LTLIBRARIES = \
  libtutorial.la

libtutorial_la_SOURCES = \
  tutorial.cc

```



```
libtutorial_la_LIBADD = \
  -lphool \
  -lSubsysReco \
  -ltrack_io

BUILT_SOURCES = testexternals.cc

noinst_PROGRAMS = \
  testexternals

testexternals_SOURCES = testexternals.cc
testexternals_LDADD   = libtutorial.la

testexternals.cc:
	echo "//*** this is a generated fi
	echo "int main()" >> $@
	echo "{" >> $@
	echo "  return 0;" >> $@
	echo "}" >> $@

clean-local:
	rm -f $(BUILT_SOURCES)
```

You need to add -ltrack_io option, which means linking libtrack_io.so to the generated file.

After changing Makefile.am, you need to run autogen.sh again.

Makefile.am generated by CreateSubsysRecoModule.pl

Compiling your analysis module: How to know a flag to be used?

ref: sample_module_2/tutorial.h, .cc

We may need to judge what should be added from the error message:

```
./.libs/libtutorial.so: undefined reference to `typeinfo for TrkrClusterContainerv4'
```

How can we do it?

After changing Makefile.am, you need to run autogen.sh again.

Makefile.am generated by CreateSubsysRecoModule.pl

Practical example2: MC, #cluster distribution

HANDS ON!

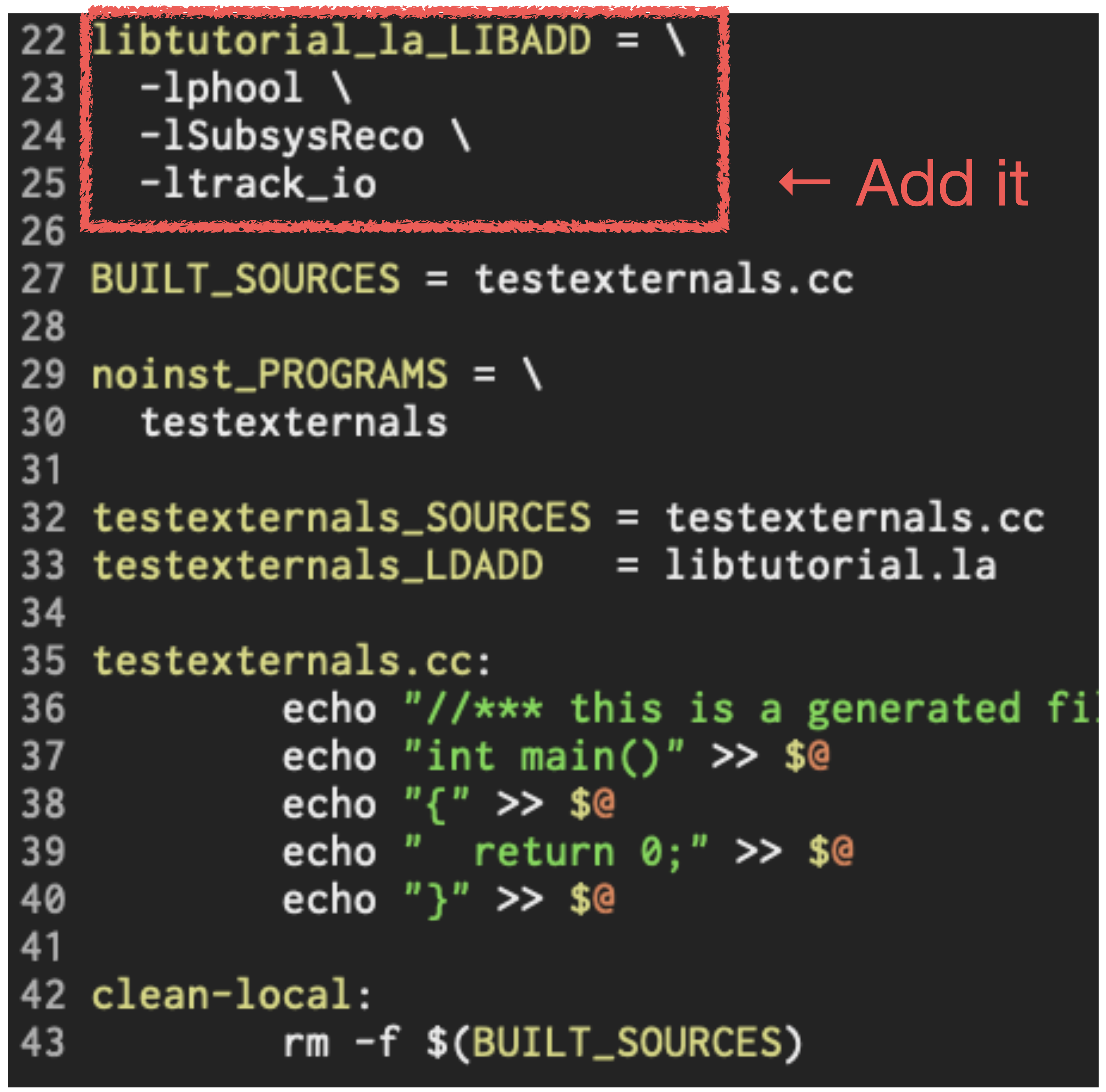
It depends on what you want to do. For example:

1. Replace tutorial.h and tutorial.cc to those in sample_module_3 (or copy&pate codes).
2. Check inside tutorial.h/.cc and find the part for
   - Open/Close the output ROOT file at the beginning/end of a run
   - Store your analysis result in a histogram
3. Modify Makefile.am
4. Execute autogen.sh again, then make and make install

```
$ cd build
$ ../autogen.sh --prefix=$PWD/../install
$ make
$ make install
```

5. Run Fun4All_minimum_3.C

```
$ root  'Fun4All_minimum_3.C( 10 )'
```

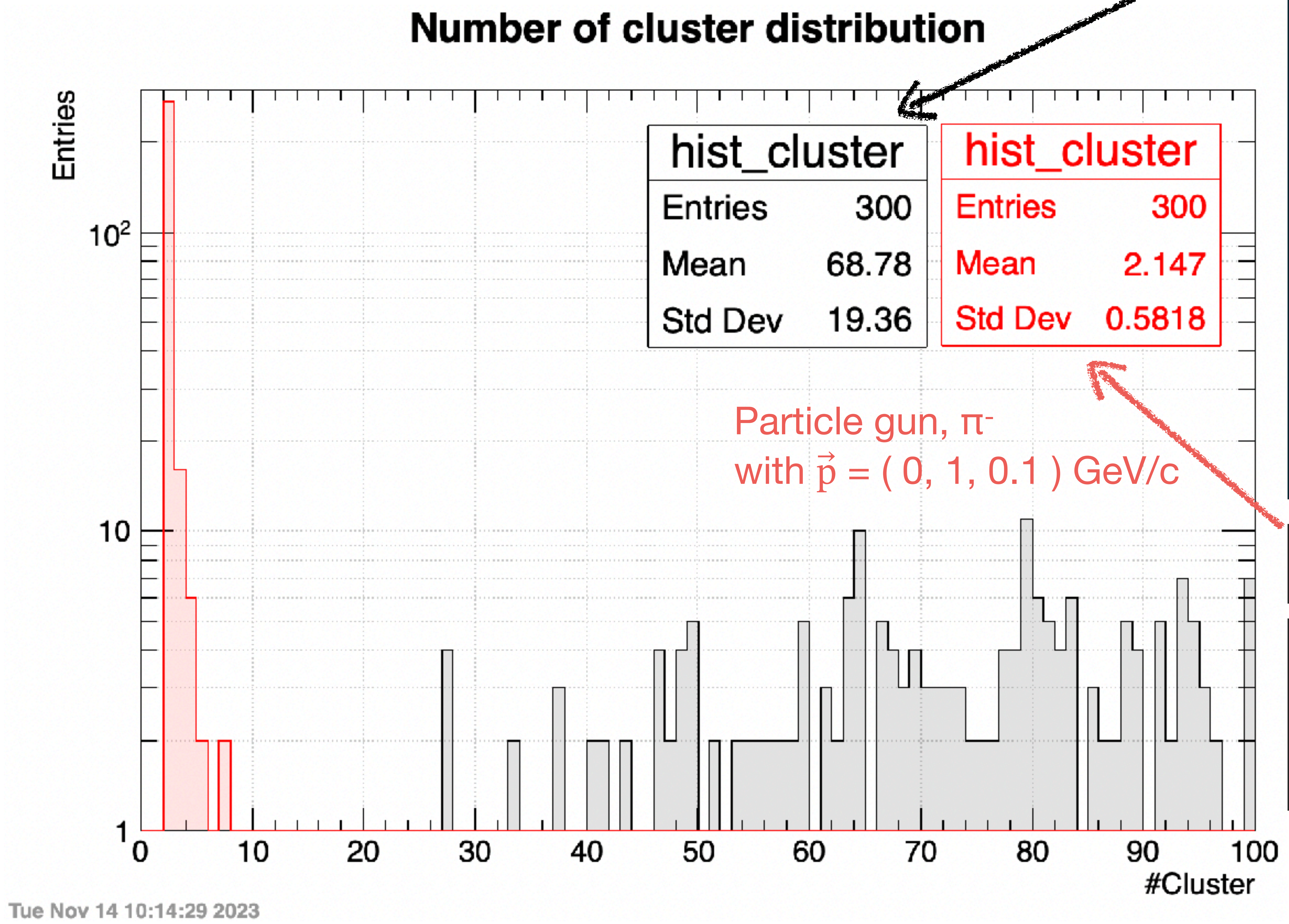


```
libtutorial_la_LIBADD = \
  -lphool \
  -lSubsysReco \
  -ltrack_io

BUILT_SOURCES = testexternals.cc

noinst_PROGRAMS = \
  testexternals

testexternals_SOURCES = testexternals.cc
testexternals_LDADD   = libtutorial.la

testexternals.cc:
	echo "//*** this is a generated fi
	echo "int main()" >> $@
	echo "{" >> $@
	echo "  return 0;" >> $@
	echo "}" >> $@

clean-local:
	rm -f $(BUILT_SOURCES)
```

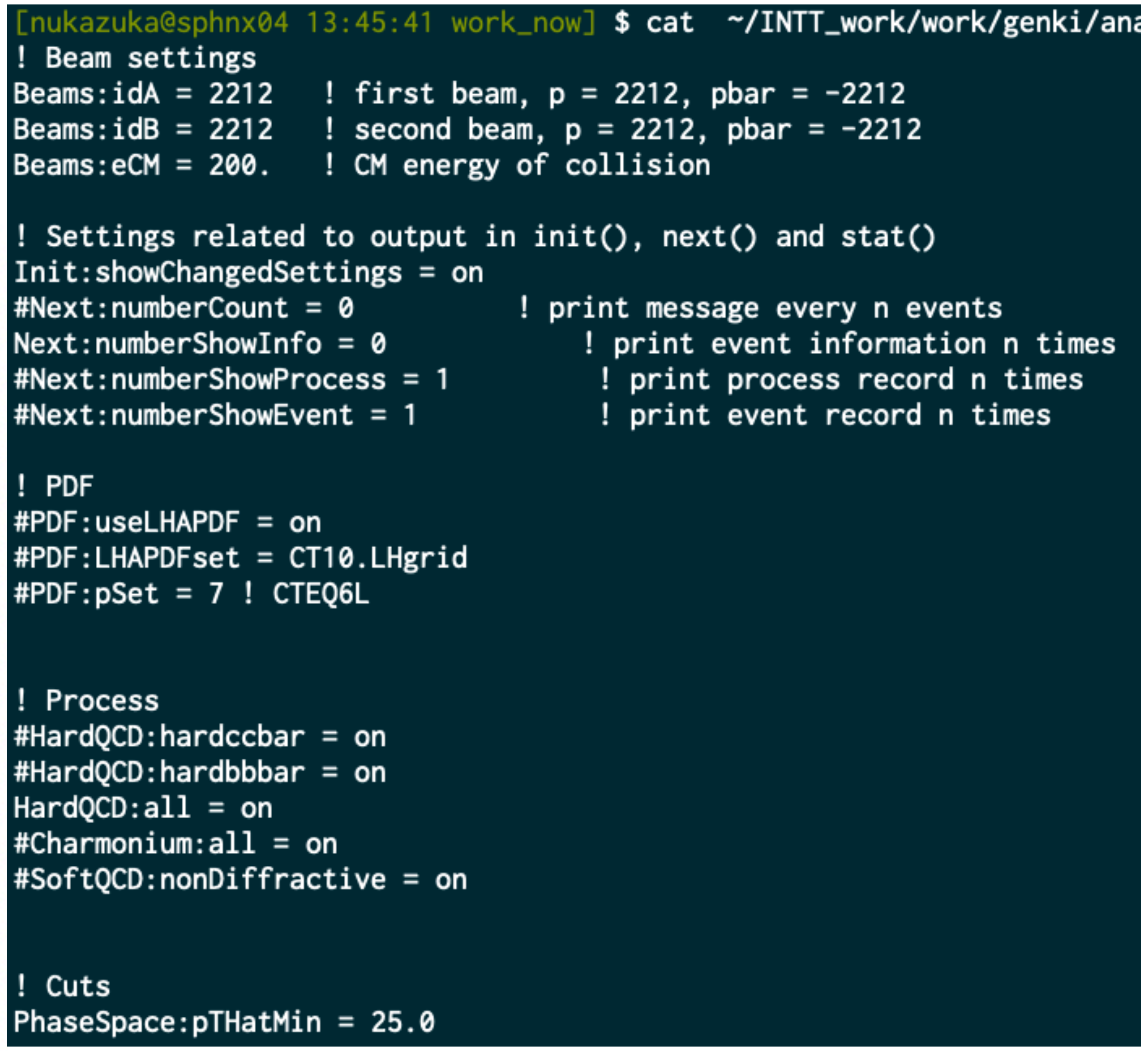
← Add it

Practical example2: MC, #cluster distribution

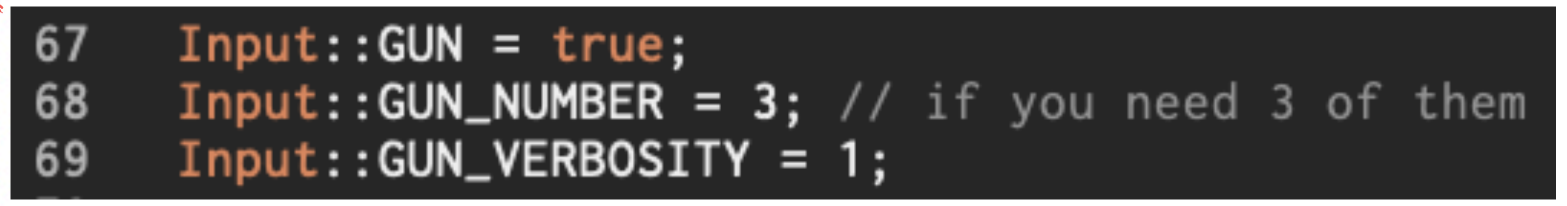
Pythia8, pp at 200 GeV



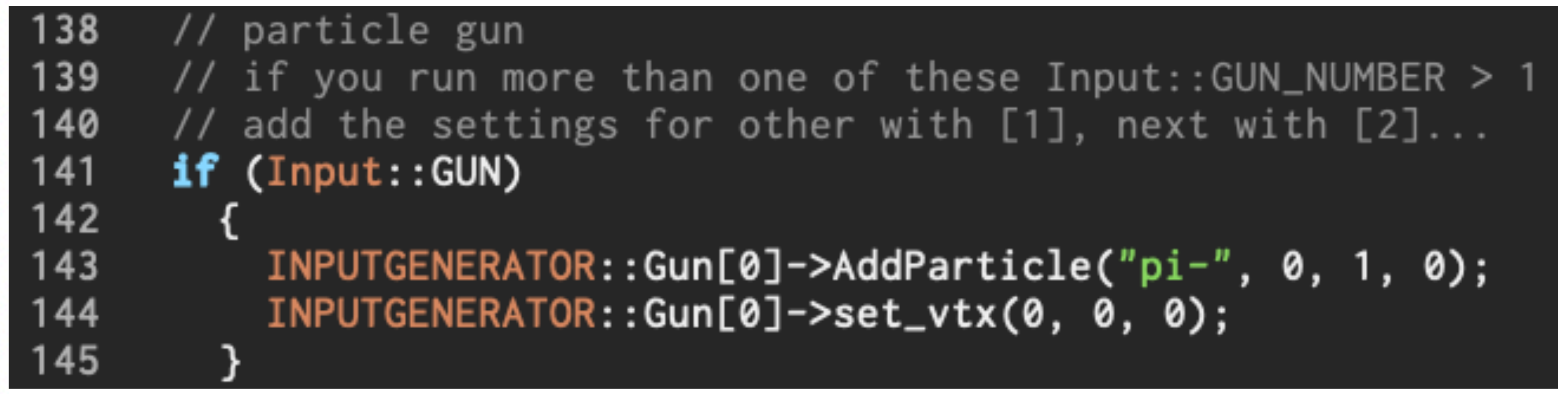
Particle gun, π⁻
with $\vec{p}$ = (0, 1, 0.1) GeV/c



```
[nukazuka@sphnx04 13:45:41 work_now] $ cat  ~/INTT_work/work/genki/ana
! Beam settings
Beams:idA = 2212    ! first beam, p = 2212, pbar = -2212
Beams:idB = 2212    ! second beam, p = 2212, pbar = -2212
Beams:eCM = 200.    ! CM energy of collision

! Settings related to output in init(), next() and stat()
Init:showChangedSettings = on
#Next:numberCount = 0           ! print message every n events
Next:numberShowInfo = 0             ! print event information n times
#Next:numberShowProcess = 1          ! print process record n times
#Next:numberShowEvent = 1            ! print event record n times

! PDF
#PDF:useLHAPDF = on
#PDF:LHAPDFset = CT10.LHgrid
#PDF:pSet = 7 ! CTEQ6L


! Process
#HardQCD:hardccbar = on
#HardQCD:hardbbbar = on
HardQCD:all = on
#Charmonium:all = on
#SoftQCD:nonDiffractive = on


! Cuts
PhaseSpace:pTHatMin = 25.0
```



```
Input::GUN = true;
Input::GUN_NUMBER = 3; // if you need 3 of them
Input::GUN_VERBOSITY = 1;
```



```
 // particle gun
 // if you run more than one of these Input::GUN_NUMBER > 1
 // add the settings for other with [1], next with [2]...
 if (Input::GUN)
  {
    INPUTGENERATOR::Gun[0]->AddParticle("pi-", 0, 1, 0);
    INPUTGENERATOR::Gun[0]->set_vtx(0, 0, 0);
  }
```

Misc

- People are interested in some commands shown in my slides but not available for you, such as “tree”. I’ll install them to `/sphenix/tg/tg01/commissioning/INTT/repositories/libraries/bin` so that you can use it by adding the path to the environment variable PATH:

  ```
   $ export PATH= /sphenix/tg/tg01/commissioning/INTT/repositories/libraries/bin:${PATH}
  ```

  If you are interested in, I can install `Emacs29, time (output is human-readable format), ag (faster than grep).`
  You can do the same. Let’s make the environment better!

- aaa

Homework

- Learn class inheritance in C++.
- Learn polymorphism.
- Learn the environment variable LD_LIBRARY_PATH
- Understand $ echo $ROOT_INCLUDE_PATH | sed -e "s/:/\n/g" | grep fun4al
- Understand
-