# CLAS12ROOT Overview

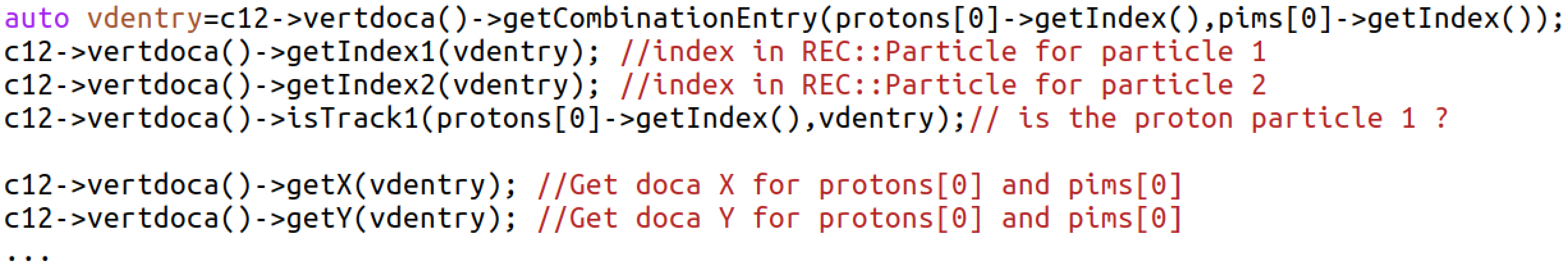
Derek Glazier

University of Glasgow

09/09/21

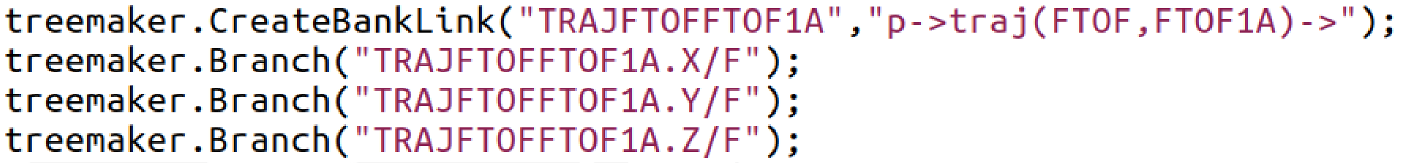
# Recent updates (up to 1.7.6)

Support for vertdoca banks



```
auto vdentry=c12->vertdoca()->getCombinationEntry(protons[0]->getIndex(),pims[0]->getIndex());
c12->vertdoca()->getIndex1(vdentry); //index in REC::Particle for particle 1
c12->vertdoca()->getIndex2(vdentry); //index in REC::Particle for particle 2
c12->vertdoca()->isTrack1(protons[0]->getIndex(),vdentry);// is the proton particle 1 ?

c12->vertdoca()->getX(vdentry); //Get doca X for protons[0] and pims[0]
c12->vertdoca()->getY(vdentry); //Get doca Y for protons[0] and pims[0]
...
```

Support for trajectories in treemaker



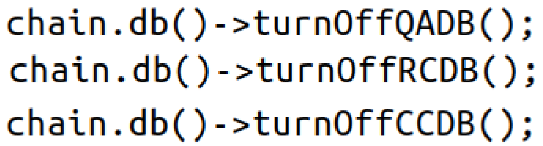
```
treemaker.CreateBankLink("TRAJFTOFFTOF1A","p->traj(FTOF,FTOF1A)->");
treemaker.Branch("TRAJFTOFFTOF1A.X/F");
treemaker.Branch("TRAJFTOFFTOF1A.Y/F");
treemaker.Branch("TRAJFTOFFTOF1A.Z/F");
```

Update clasQADB to v1.2.0

Fix some issues with clas12root::qadb and simulated data

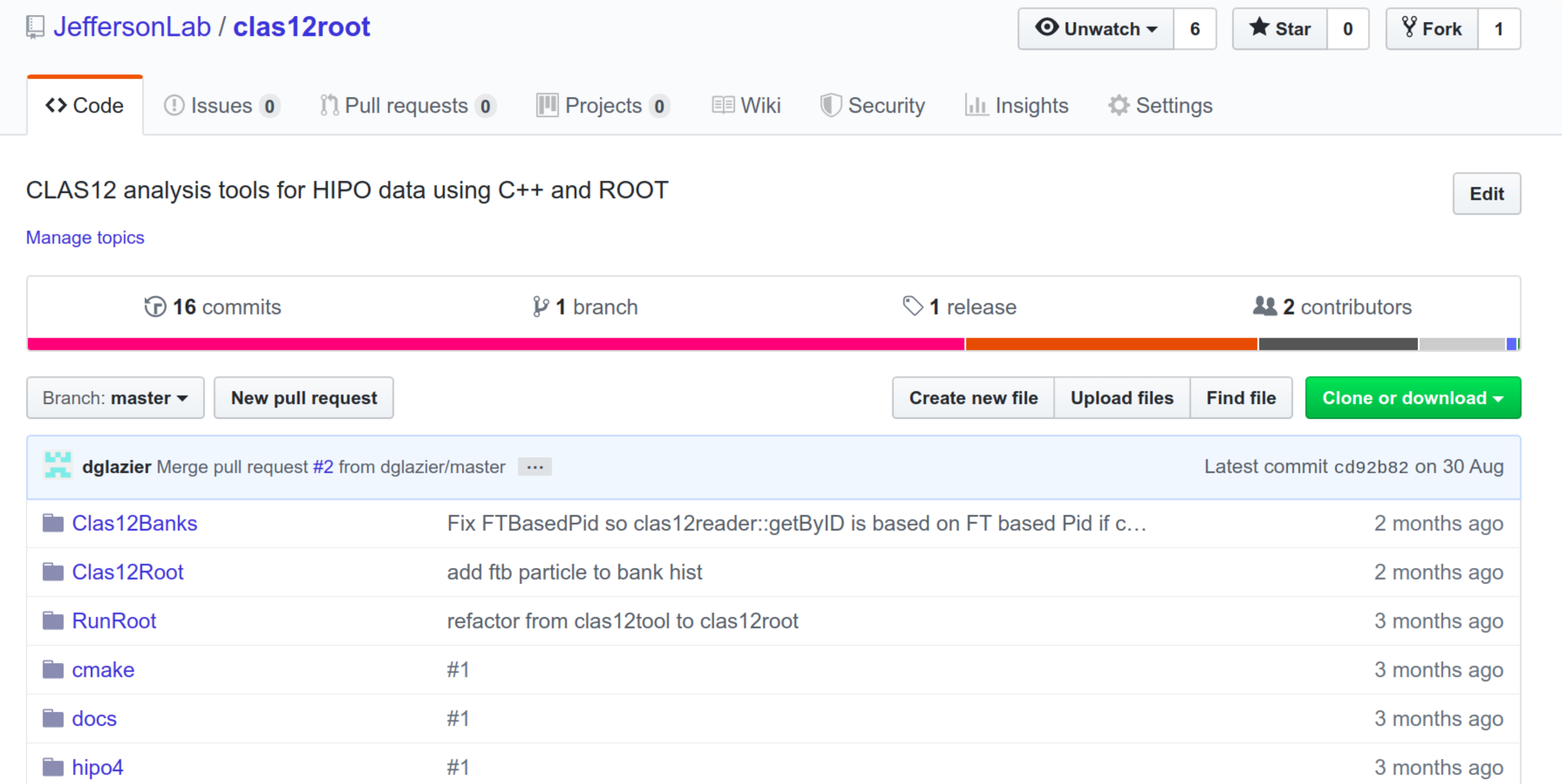
Note HipoSelector can not give correct accumulated charge

Allow to turn off databases



```
chain.db()->turnOffQADB();
chain.db()->turnOffRCDB();
chain.db()->turnOffCCDB();
```

# CLAS12ROOT @JeffersonLab Github

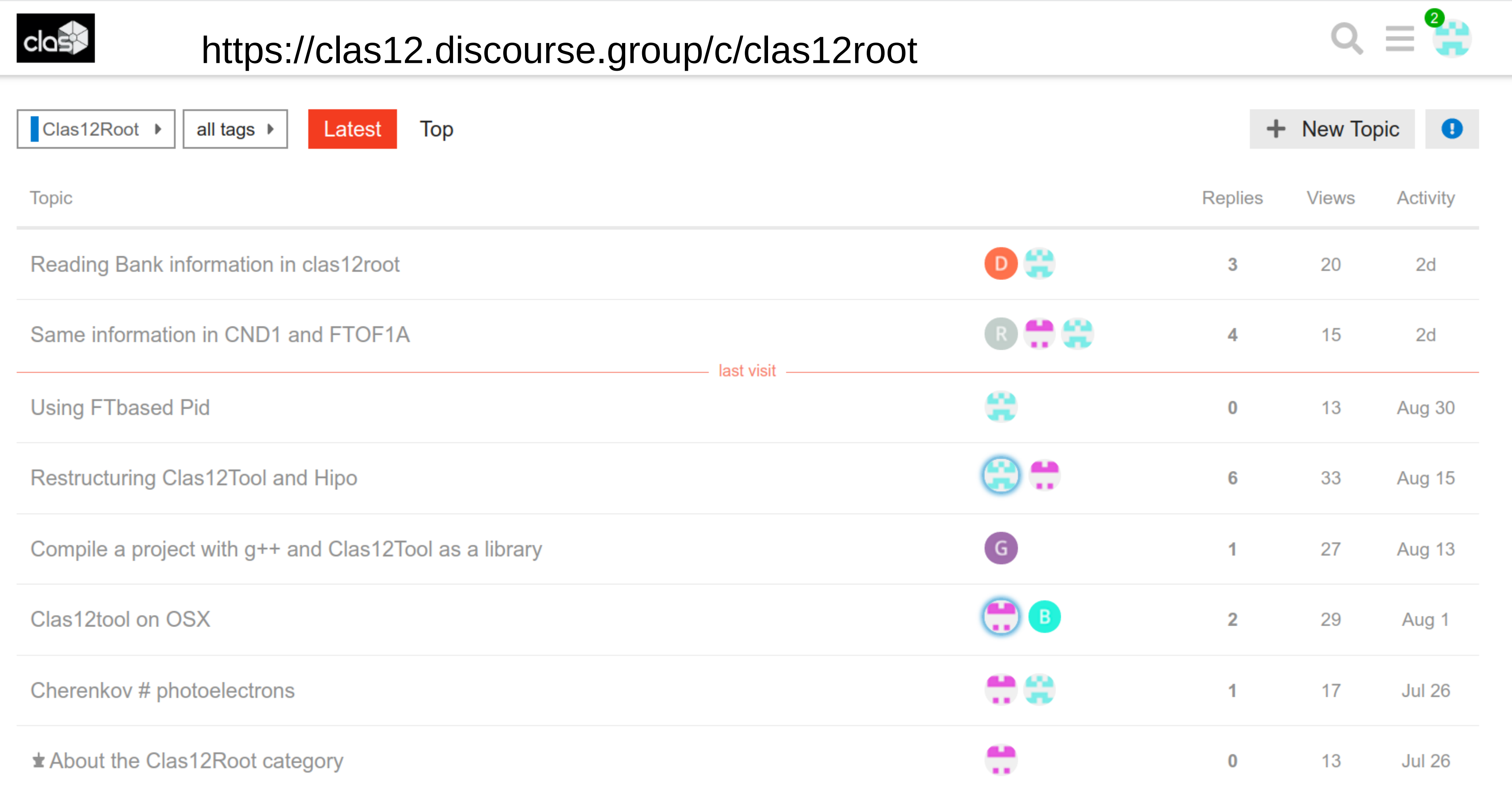


CLAS12 analysis tools for HIPO data using C++ and ROOT

dglazier Merge pull request #2 from dglazier/master … Latest commit cd92b82 on 30 Aug

| | | |
|---|---|---|
| Clas12Banks | Fix FTBasedPid so clas12reader::getByID is based on FT based Pid if c… | 2 months ago |
| Clas12Root | add ftb particle to bank hist | 2 months ago |
| RunRoot | refactor from clas12tool to clas12root | 3 months ago |
| cmake | #1 | 3 months ago |
| docs | #1 | 3 months ago |
| hipo4 | #1 | 3 months ago |

# CLAS12ROOT @JeffersonLab Forum

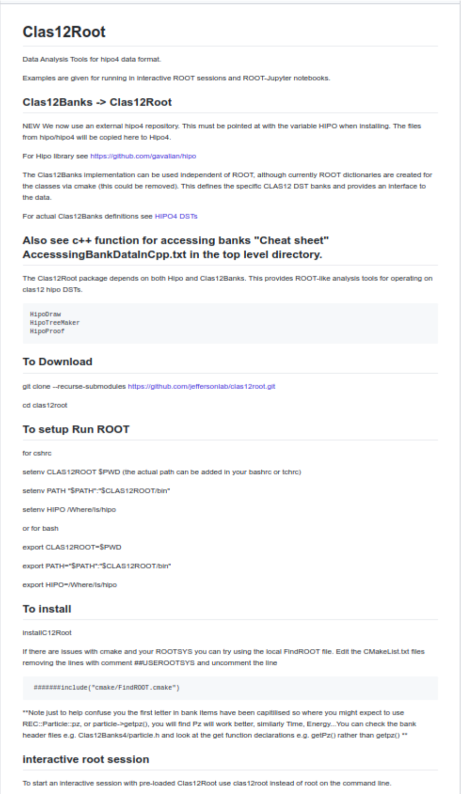


https://clas12.discourse.group/c/clas12root

| Topic | Replies | Views | Activity |
|---|---|---|---|
| Reading Bank information in clas12root | 3 | 20 | 2d |
| Same information in CND1 and FTOF1A | 4 | 15 | 2d |
| Using FTbased Pid | 0 | 13 | Aug 30 |
| Restructuring Clas12Tool and Hipo | 6 | 33 | Aug 15 |
| Compile a project with g++ and Clas12Tool as a library | 1 | 27 | Aug 13 |
| Clas12tool on OSX | 2 | 29 | Aug 1 |
| Cherenkov # photoelectrons | 1 | 17 | Jul 26 |
| About the Clas12Root category | 0 | 13 | Jul 26 |

last visit

# CLAS12ROOT @JeffersonLab Instructions



## Clas12Root

Data Analysis Tools for hipo4 data format.

Examples are given for running in interactive ROOT sessions and ROOT-Jupyter notebooks.

### Clas12Banks -> Clas12Root

NEW We now use an external hipo4 repository. This must be pointed at with the variable HIPO when installing. The files from hipo/hipo4 will be copied here to Hipo4.

For Hipo library see https://github.com/gavalian/hipo

The Clas12Banks implementation can be used independent of ROOT, although currently ROOT dictionaries are created for the classes via cmake (this could be removed). This defines the specific CLAS12 DST banks and provides an interface to the data.

For actual Clas12Banks definitions see HIPO4 DSTs

### Also see c++ function for accessing banks "Cheat sheet" AccesssingBankDataInCpp.txt in the top level directory.

The Clas12Root package depends on both Hipo and Clas12Banks. This provides ROOT-like analysis tools for operating on clas12 hipo DSTs.

```
HipoDraw
HipoTreeMaker
HipoProof
```

### To Download

git clone --recurse-submodules https://github.com/jeffersonlab/clas12root.git

cd clas12root

### To setup Run ROOT

for cshrc

setenv CLAS12ROOT $PWD (the actual path can be added in your bashrc or tchrc)

setenv PATH "$PATH":"$CLAS12ROOT/bin"

setenv HIPO /Where/Is/hipo

or for bash

export CLAS12ROOT=$PWD

export PATH="$PATH":"$CLAS12ROOT/bin"

export HIPO=/Where/Is/hipo

### To install

installC12Root

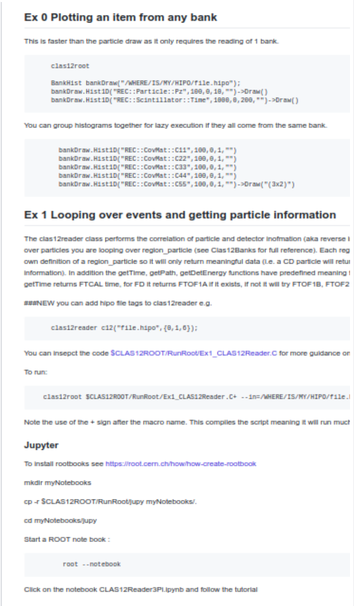
If there are issues with cmake and your ROOTSYS you can try using the local FindROOT file. Edit the CMakeList.txt files removing the lines with comment ##USEROOTSYS and uncomment the line

```
#######include("cmake/FindROOT.cmake")
```

**Note just to help confuse you the first letter in bank items have been capitilised so where you might expect to use REC::Particle::pz, or particle->getpz(), you will find Pz will work better, similarly Time, Energy...You can check the bank header files e.g. Clas12Banks4/particle.h and look at the get function declarations e.g. getPz() rather than getpz() **

### interactive root session

To start an interactive session with pre-loaded Clas12Root use clas12root instead of root on the command line.



### Ex 0 Plotting an item from any bank

This is faster than the particle draw as it only requires the reading of 1 bank.

```
    clas12root

    BankHist bankDraw("/WHERE/IS/MY/HIPO/file.hipo");
    bankDraw.Hist1D("REC::Particle::Pz",100,0,10,"")->Draw()
    bankDraw.Hist1D("REC::Scintillator::Time",1000,0,200,"")->Draw()
```

You can group histograms together for lazy execution if they all come from the same bank.

```
     bankDraw.Hist1D("REC::CovMat::C11",100,0,1,"")
     bankDraw.Hist1D("REC::CovMat::C22",100,0,1,"")
     bankDraw.Hist1D("REC::CovMat::C33",100,0,1,"")
     bankDraw.Hist1D("REC::CovMat::C44",100,0,1,"")
     bankDraw.Hist1D("REC::CovMat::C55",100,0,1,"")->Draw("(3x2)")
```

### Ex 1 Looping over events and getting particle information

The clas12reader class performs the correlation of particle and detector inofmation (aka reverse i over particles you are looping over region_particle (see Clas12Banks for full reference). Each reg own definition of a region_particle so it will only return meaningful data (i.e. a CD particle will retu information). In addition the getTime, getPath, getDetEnergy functions have predefined meaning getTime returns FTCAL time, for FD it returns FTOF1A if it exists, if not it will try FTOF1B, FTOF2

###NEW you can add hipo file tags to clas12reader e.g.

```
    clas12reader c12("file.hipo",{0,1,6});
```

You can insepct the code $CLAS12ROOT/RunRoot/Ex1_CLAS12Reader.C for more guidance on

To run:

```
   clas12root $CLAS12ROOT/RunRoot/Ex1_CLAS12Reader.C+ --in=/WHERE/IS/MY/HIPO/file.
```

Note the use of the + sign after the macro name. This compiles the script meaning it will run muc

#### Jupyter

To install rootbooks see https://root.cern.ch/how/how-create-rootbook

mkdir myNotebooks

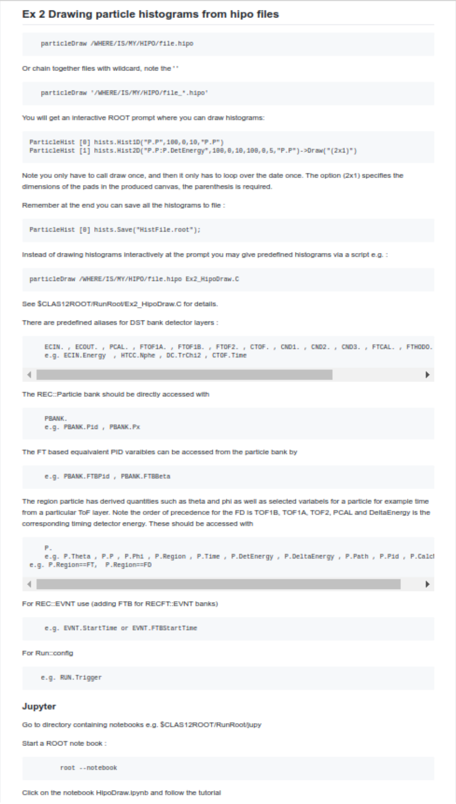
cp -r $CLAS12ROOT/RunRoot/jupy myNotebooks/.

cd myNotebooks/jupy

Start a ROOT note book :

```
        root --notebook
```

Click on the notebook CLAS12Reader3Pi.ipynb and follow the tutorial



### Ex 2 Drawing particle histograms from hipo files

```
   particleDraw /WHERE/IS/MY/HIPO/file.hipo
```

Or chain together files with wildcard, note the ''

```
   particleDraw '/WHERE/IS/MY/HIPO/file_*.hipo'
```

You will get an interactive ROOT prompt where you can draw histograms:

```
ParticleHist [0] hists.Hist1D("P.P",100,0,10,"P.P")
ParticleHist [1] hists.Hist2D("P.P:P.DetEnergy",100,0,10,100,0,5,"P.P")->Draw("(2x1)")
```

Note you only have to call draw once, and then it only has to loop over the date once. The option (2x1) specifies the dimensions of the pads in the produced canvas, the parenthesis is required.

Remember at the end you can save all the histograms to file :

```
ParticleHist [0] hists.Save("HistFile.root");
```

Instead of drawing histograms interactively at the prompt you may give predefined histograms via a script e.g. :

```
particleDraw /WHERE/IS/MY/HIPO/file.hipo Ex2_HipoDraw.C
```

See $CLAS12ROOT/RunRoot/Ex2_HipoDraw.C for details.

There are predefined aliases for DST bank detector layers :

```
    ECIN. , ECOUT. , PCAL. , FTOF1A. , FTOF1B. , FTOF2. , CTOF. , CND1. , CND2. , CND3. , FTCAL. , FTHODO.
    e.g. ECIN.Energy  , HTCC.Nphe , DC.TrChi2 , CTOF.Time
```

The REC::Particle bank should be directly accessed with

```
    PBANK.
    e.g. PBANK.Pid , PBANK.Px
```

The FT based equaivalent PID varaibles can be accessed from the particle bank by

```
    e.g. PBANK.FTBPid , PBANK.FTBBeta
```

The region particle has derived quantities such as theta and phi as well as selected variabels for a particle for example time from a particular ToF layer. Note the order of precedence for the FD is TOF1B, TOF1A, TOF2, PCAL and DeltaEnergy is the corresponding timing detector energy. These should be accessed with

```
    P.
    e.g. P.Theta , P.P , P.Phi , P.Region , P.Time , P.DetEnergy , P.DeltaEnergy , P.Path , P.Pid , P.Calc
e.g. P.Region==FT,  P.Region==FD
```

For REC::EVNT use (adding FTB for RECFT::EVNT banks)

```
    e.g. EVNT.StartTime or EVNT.FTBStartTime
```

For Run::config

```
   e.g. RUN.Trigger
```

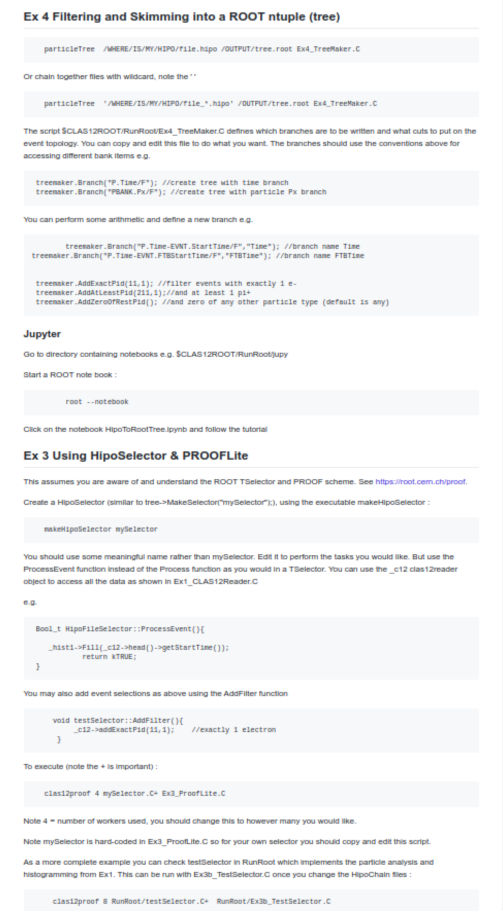
#### Jupyter

Go to directory containing notebooks e.g. $CLAS12ROOT/RunRoot/jupy

Start a ROOT note book :

```
        root --notebook
```

Click on the notebook HipoDraw.ipynb and follow the tutorial



### Ex 4 Filtering and Skimming into a ROOT ntuple (tree)

```
   particleTree  /WHERE/IS/MY/HIPO/file.hipo /OUTPUT/tree.root Ex4_TreeMaker.C
```

Or chain together files with wildcard, note the ''

```
   particleTree  '/WHERE/IS/MY/HIPO/file_*.hipo' /OUTPUT/tree.root Ex4_TreeMaker.C
```

The script $CLAS12ROOT/RunRoot/Ex4_TreeMaker.C defines which branches are to be written and what cuts to put on the event topology. You can copy and edit this file to do what you want. The branches should use the conventions above for accessing different bank items e.g.

```
  treemaker.Branch("P.Time/F"); //create tree with time branch
  treemaker.Branch("PBANK.Px/F"); //create tree with particle Px branch
```

You can perform some arithmetic and define a new branch e.g.

```
        treemaker.Branch("P.Time-EVNT.StartTime/F","Time"); //branch name Time
 treemaker.Branch("P.Time-EVNT.FTBStartTime/F","FTBTime"); //branch name FTBTime


  treemaker.AddExactPid(11,1); //filter events with exactly 1 e-
  treemaker.AddAtLeastPid(211,1);//and at least 1 pi+
  treemaker.AddZeroOfRestPid(); //and zero of any other particle type (default is any)
```

#### Jupyter

Go to directory containing notebooks e.g. $CLAS12ROOT/RunRoot/jupy

Start a ROOT note book :

```
        root --notebook
```

Click on the notebook HipoToRootTree.ipynb and follow the tutorial

### Ex 3 Using HipoSelector & PROOFLite

This assumes you are aware of and understand the ROOT TSelector and PROOF scheme. See https://root.cern.ch/proof.

Create a HipoSelector (similar to tree->MakeSelector("mySelector");), using the executable makeHipoSelector :

```
   makeHipoSelector mySelector
```

You should use some meaningful name rather than mySelector. Edit it to perform the tasks you would like. But use the ProcessEvent function instead of the Process function as you would in a TSelector. You can use the _c12 clas12reader object to access all the data as shown in Ex1_CLAS12Reader.C

e.g.

```
  Bool_t HipoFileSelector::ProcessEvent(){

     _hist1->Fill(_c12->head()->getStartTime());
            return kTRUE;
  }
```

You may also add event selections as above using the AddFilter function

```
     void testSelector::AddFilter(){
         _c12->addExactPid(11,1);    //exactly 1 electron
      }
```

To execute (note the + is important) :

```
   clas12proof 4 mySelector.C+ Ex3_ProofLite.C
```

Note 4 = number of workers used, you should change this to however many you would like.

Note mySelector is hard-coded in Ex3_ProofLite.C so for your own selector you should copy and edit this script.

As a more complete example you can check testSelector in RunRoot which implements the particle analysis and histogramming from Ex1. This can be run with Ex3b_TestSelector.C once you change the HipoChain files :

```
     clas12proof 8 RunRoot/testSelector.C+  RunRoot/Ex3b_TestSelector.C
```

# Clas12root Download

## See README at https://github.com/JeffersonLab/clas12root

```
git clone -recurse-submodules https://github.com/jeffersonlab/clas12root.git

cd clas12root

git checkout v1.7

setenv CLAS12ROOT $PWD

setenv PATH "$PATH":"$CLAS12ROOT/bin"

#To use the RCDB interface

setenv RCDB_HOME /Where/Is/rcdb (e.g. setenv RCDB_HOME
                                      ${CLAS12ROOT}/rcdb )

#To use the CCDB interface

setenv CCDB_HOME /Where/Is/ccdb

#To use clasqaDB interface

setenv QADB /Where/Is/clasqaDB

./installC12Root

Note, if the DBs are not set it will compile without
```

i.e local ccdb
cloned as submodule

```
cd $CCDB_HOME
source environment.csh
scons
Cd $CLAS12ROOT
```

# CLAS12ROOT on ifarm

```
source /group/clas12/packages/setup.csh
(or setup.sh for bash)

module load clas12/pro or dev

OR module switch clas12root/1.7.6 etc
```

# CLAS12ROOT Goals

Provide access to hipo file DST data

...and run databases

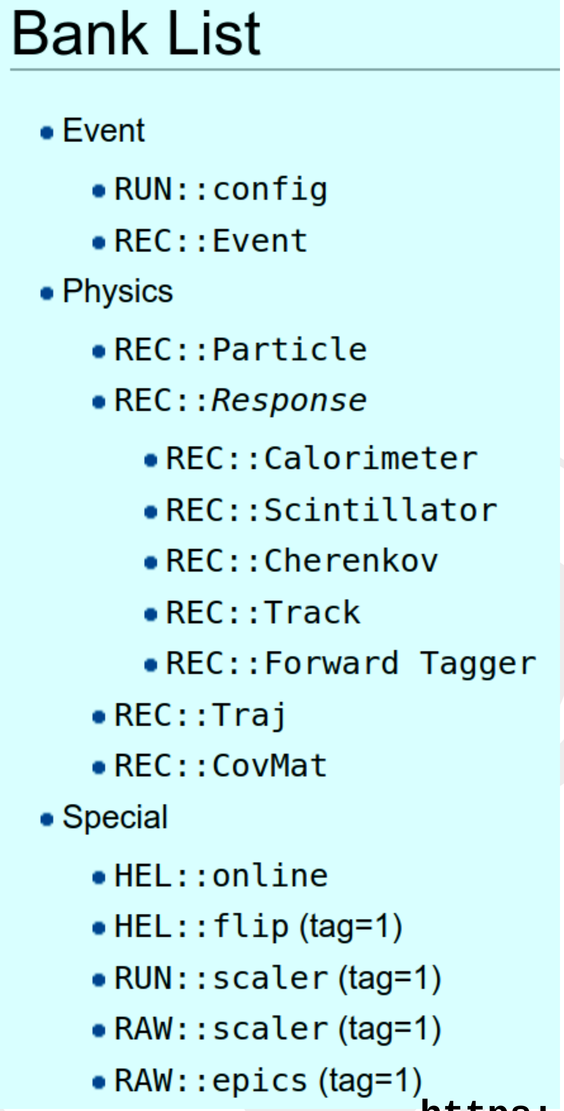
Links detector data to particles

Runs like ROOT script

Add other ROOT like tools

particleDraw, HipoSelector,...

# Why CLAS12ROOT ?  HiPO DSTs



## Bank List

- Event
  - RUN::config
  - REC::Event
- Physics
  - REC::Particle
  - REC::*Response*
    - REC::Calorimeter
    - REC::Scintillator
    - REC::Cherenkov
    - REC::Track
    - REC::Forward Tagger
  - REC::Traj
  - REC::CovMat
- Special
  - HEL::online
  - HEL::flip (tag=1)
  - RUN::scaler (tag=1)
  - RAW::scaler (tag=1)
  - RAW::epics (tag=1)

Data packed into different banks

Each bank can have a number of entries
e.g. REC::Particle::px, REC::Particle::Pid
each with the same number of values per event

Different bank can have a different number of values per event

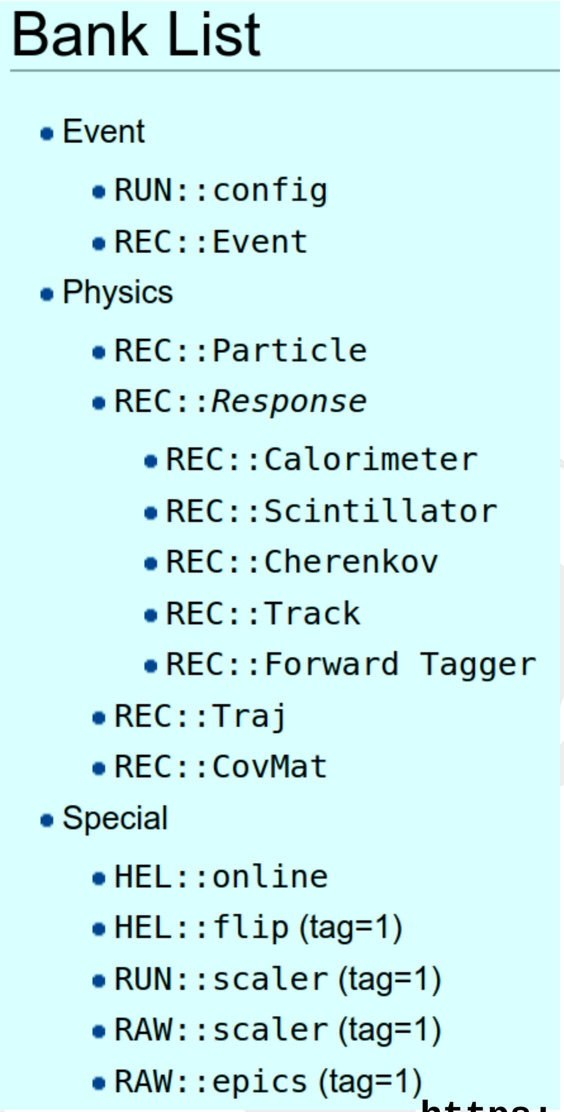
“No” redundant information saved
→ Minimise file size

e.g. if there are 6 particles but only 2 scintillator hits, REC::Scintillator Will contain only 2 values per entry

Note tag=1 banks only written at helcity flip. Many fewer tag=1 events

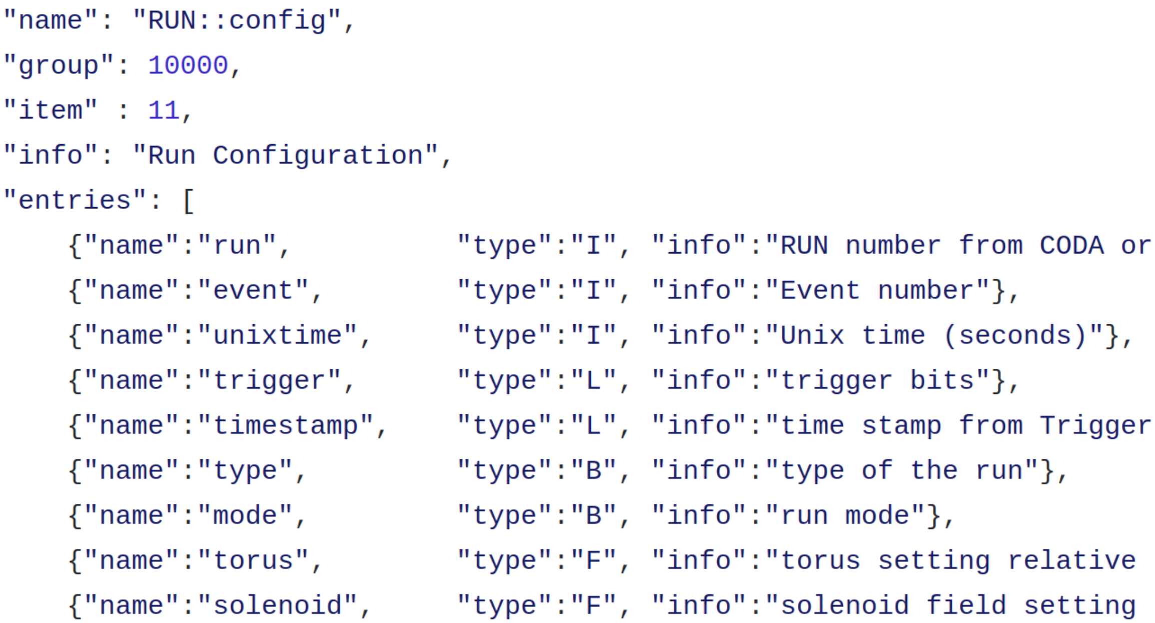
https://clasweb.jlab.org/wiki/index.php/CLAS12_DSTs

# Why CLAS12ROOT ?  HiPO DSTs



## Bank List

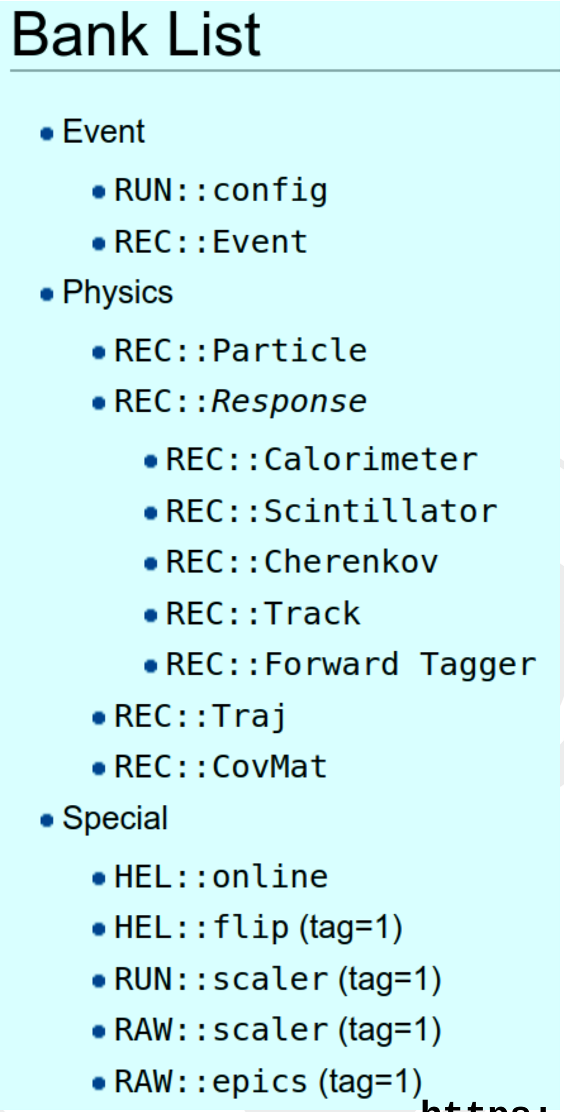
- Event
  - RUN::config
  - REC::Event
- Physics
  - REC::Particle
  - REC::*Response*
    - REC::Calorimeter
    - REC::Scintillator
    - REC::Cherenkov
    - REC::Track
    - REC::Forward Tagger
  - REC::Traj
  - REC::CovMat
- Special
  - HEL::online
  - HEL::flip (tag=1)
  - RUN::scaler (tag=1)
  - RAW::scaler (tag=1)
  - RAW::epics (tag=1)



```
"name": "RUN::config",
"group": 10000,
"item" : 11,
"info": "Run Configuration",
"entries": [
    {"name":"run",          "type":"I", "info":"RUN number from CODA or
    {"name":"event",        "type":"I", "info":"Event number"},
    {"name":"unixtime",     "type":"I", "info":"Unix time (seconds)"},
    {"name":"trigger",      "type":"L", "info":"trigger bits"},
    {"name":"timestamp",    "type":"L", "info":"time stamp from Trigger
    {"name":"type",         "type":"B", "info":"type of the run"},
    {"name":"mode",         "type":"B", "info":"run mode"},
    {"name":"torus",        "type":"F", "info":"torus setting relative
    {"name":"solenoid",     "type":"F", "info":"solenoid field setting
```

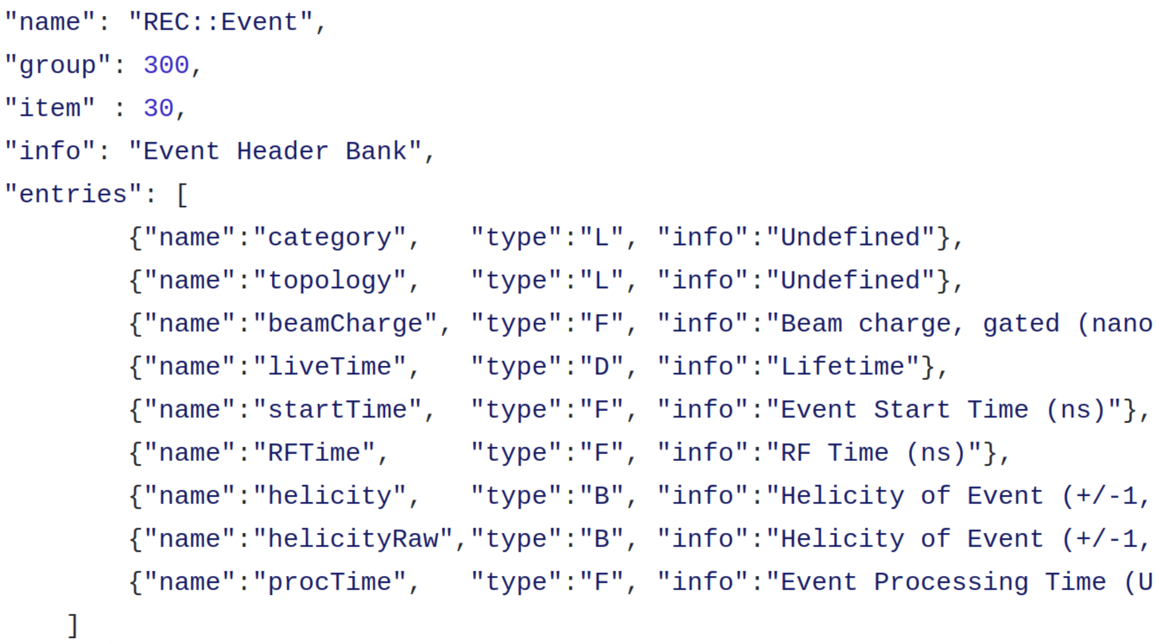
https://clasweb.jlab.org/wiki/index.php/CLAS12_DSTs

# Why CLAS12ROOT ?  HiPO DSTs



## Bank List

- Event
  - RUN::config
  - REC::Event
- Physics
  - REC::Particle
  - REC::*Response*
    - REC::Calorimeter
    - REC::Scintillator
    - REC::Cherenkov
    - REC::Track
    - REC::Forward Tagger
  - REC::Traj
  - REC::CovMat
- Special
  - HEL::online
  - HEL::flip (tag=1)
  - RUN::scaler (tag=1)
  - RAW::scaler (tag=1)
  - RAW::epics (tag=1)

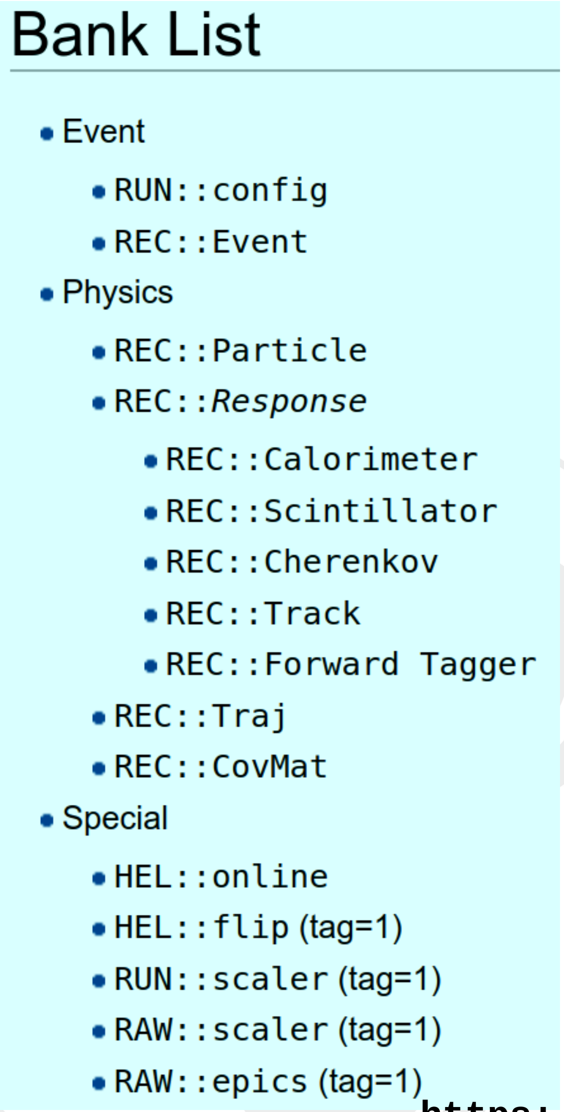


```
"name": "REC::Event",
"group": 300,
"item" : 30,
"info": "Event Header Bank",
"entries": [
        {"name":"category",   "type":"L", "info":"Undefined"},
        {"name":"topology",   "type":"L", "info":"Undefined"},
        {"name":"beamCharge", "type":"F", "info":"Beam charge, gated (nano
        {"name":"liveTime",   "type":"D", "info":"Lifetime"},
        {"name":"startTime",  "type":"F", "info":"Event Start Time (ns)"},
        {"name":"RFTime",     "type":"F", "info":"RF Time (ns)"},
        {"name":"helicity",   "type":"B", "info":"Helicity of Event (+/-1,
        {"name":"helicityRaw","type":"B", "info":"Helicity of Event (+/-1,
        {"name":"procTime",   "type":"F", "info":"Event Processing Time (U
    ]
```

Recommend : Use beamCharge from clasQADB
            not REC::Event::beamCharge
            see clas12root::qadb

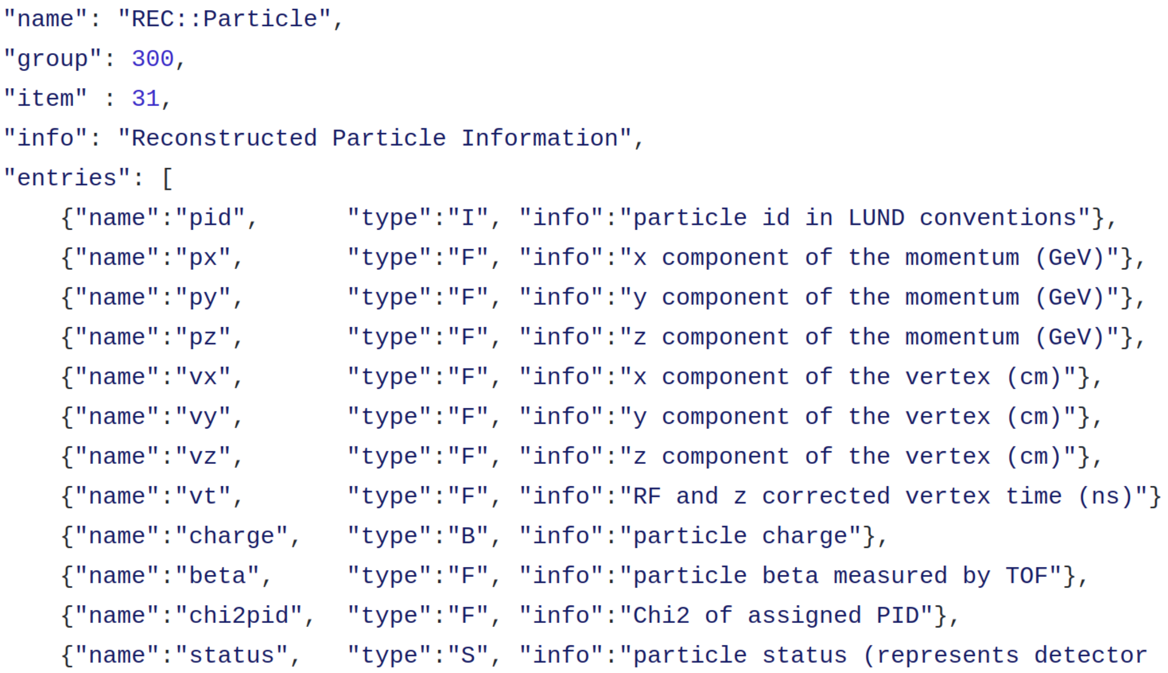
https://clasweb.jlab.org/wiki/index.php/CLAS12_DSTs

# Why CLAS12ROOT ?  HiPO DSTs



## Bank List

- Event
  - RUN::config
  - REC::Event
- Physics
  - REC::Particle
  - REC::*Response*
    - REC::Calorimeter
    - REC::Scintillator
    - REC::Cherenkov
    - REC::Track
    - REC::Forward Tagger
  - REC::Traj
  - REC::CovMat
- Special
  - HEL::online
  - HEL::flip (tag=1)
  - RUN::scaler (tag=1)
  - RAW::scaler (tag=1)
  - RAW::epics (tag=1)

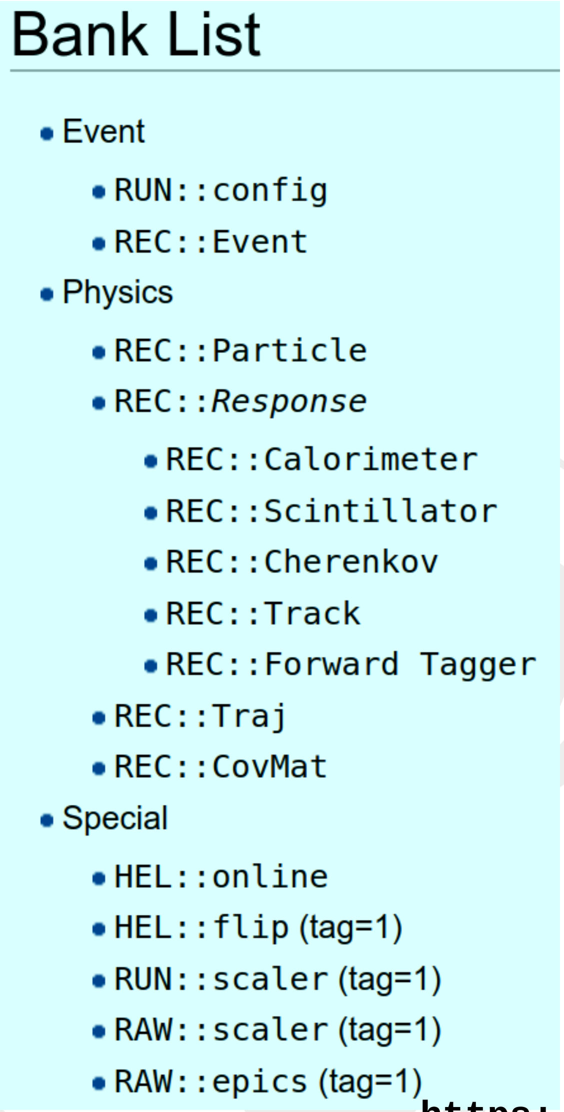


```
"name": "REC::Particle",
"group": 300,
"item" : 31,
"info": "Reconstructed Particle Information",
"entries": [
    {"name":"pid",      "type":"I", "info":"particle id in LUND conventions"},
    {"name":"px",       "type":"F", "info":"x component of the momentum (GeV)"},
    {"name":"py",       "type":"F", "info":"y component of the momentum (GeV)"},
    {"name":"pz",       "type":"F", "info":"z component of the momentum (GeV)"},
    {"name":"vx",       "type":"F", "info":"x component of the vertex (cm)"},
    {"name":"vy",       "type":"F", "info":"y component of the vertex (cm)"},
    {"name":"vz",       "type":"F", "info":"z component of the vertex (cm)"},
    {"name":"vt",       "type":"F", "info":"RF and z corrected vertex time (ns)"}
    {"name":"charge",   "type":"B", "info":"particle charge"},
    {"name":"beta",     "type":"F", "info":"particle beta measured by TOF"},
    {"name":"chi2pid",  "type":"F", "info":"Chi2 of assigned PID"},
    {"name":"status",   "type":"S", "info":"particle status (represents detector
```

Simplest analysis just needs REC::Particle
1 entry per reconstructed particle

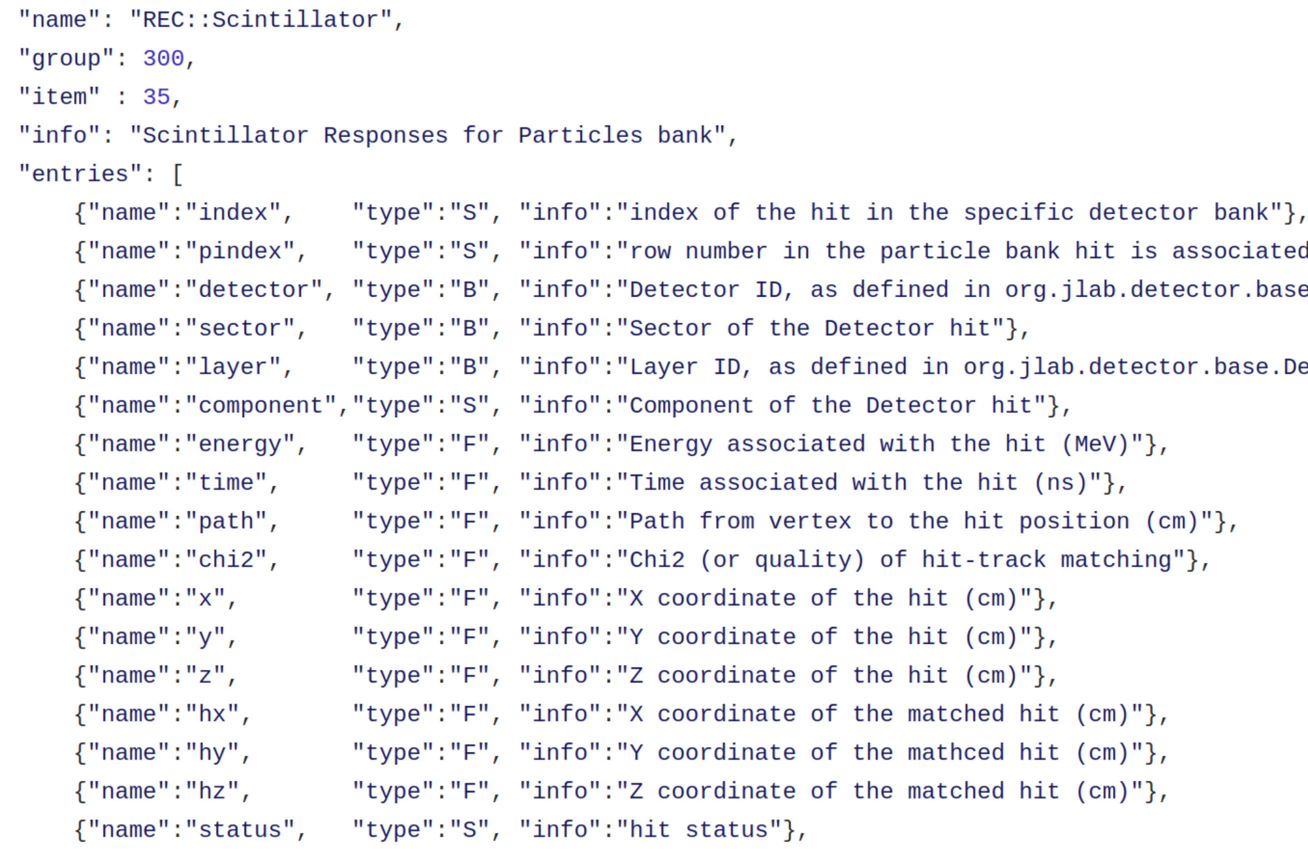
https://clasweb.jlab.org/wiki/index.php/CLAS12_DSTs

# Why CLAS12ROOT ?  HiPO DSTs



## Bank List

- Event
  - RUN::config
  - REC::Event
- Physics
  - REC::Particle
  - REC::*Response*
    - REC::Calorimeter
    - REC::Scintillator
    - REC::Cherenkov
    - REC::Track
    - REC::Forward Tagger
  - REC::Traj
  - REC::CovMat
- Special
  - HEL::online
  - HEL::flip (tag=1)
  - RUN::scaler (tag=1)
  - RAW::scaler (tag=1)
  - RAW::epics (tag=1)



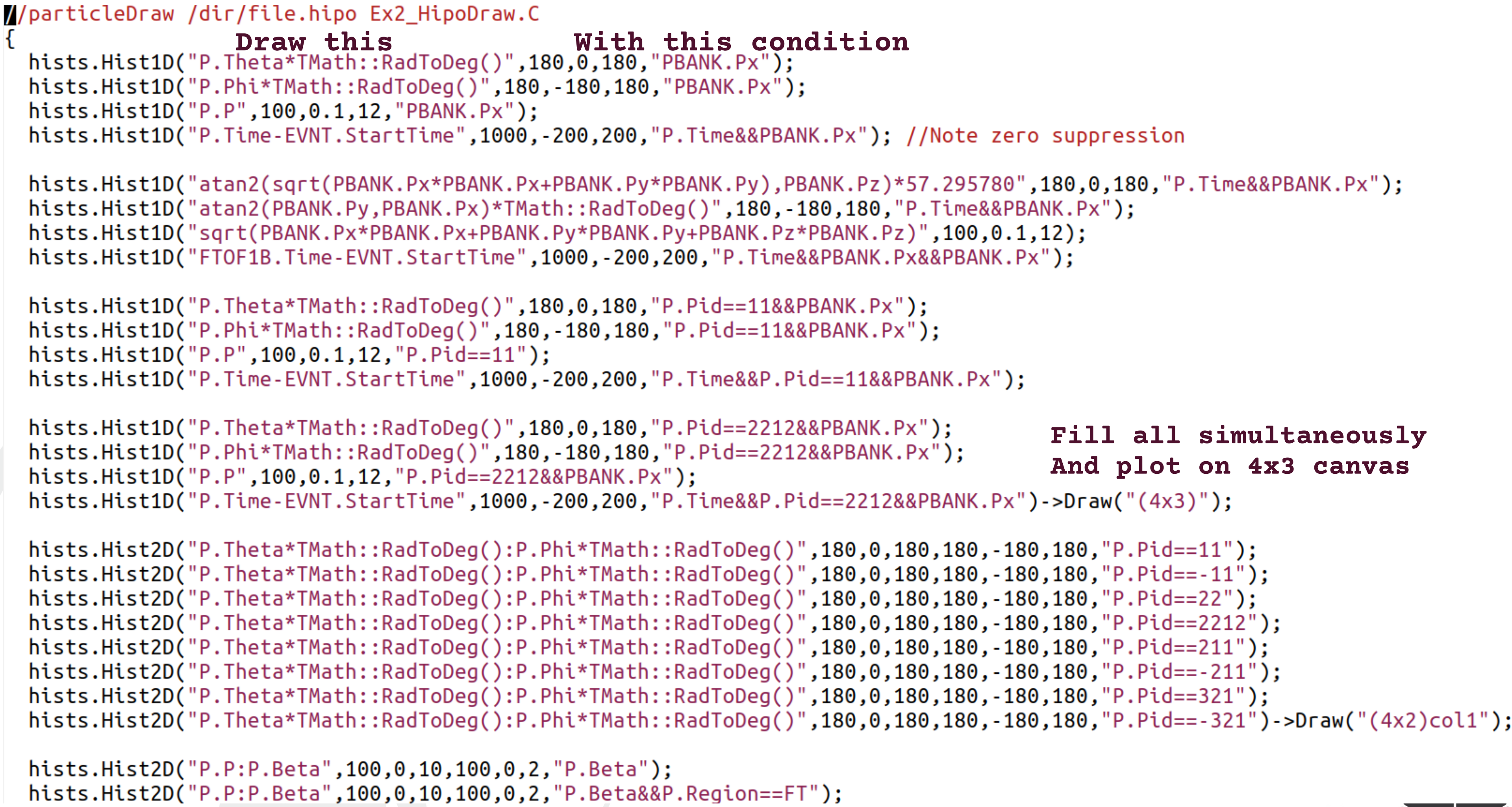
```
"name": "REC::Scintillator",
"group": 300,
"item" : 35,
"info": "Scintillator Responses for Particles bank",
"entries": [
    {"name":"index",    "type":"S", "info":"index of the hit in the specific detector bank"},
    {"name":"pindex",   "type":"S", "info":"row number in the particle bank hit is associated
    {"name":"detector", "type":"B", "info":"Detector ID, as defined in org.jlab.detector.base
    {"name":"sector",   "type":"B", "info":"Sector of the Detector hit"},
    {"name":"layer",    "type":"B", "info":"Layer ID, as defined in org.jlab.detector.base.De
    {"name":"component","type":"S", "info":"Component of the Detector hit"},
    {"name":"energy",   "type":"F", "info":"Energy associated with the hit (MeV)"},
    {"name":"time",     "type":"F", "info":"Time associated with the hit (ns)"},
    {"name":"path",     "type":"F", "info":"Path from vertex to the hit position (cm)"},
    {"name":"chi2",     "type":"F", "info":"Chi2 (or quality) of hit-track matching"},
    {"name":"x",        "type":"F", "info":"X coordinate of the hit (cm)"},
    {"name":"y",        "type":"F", "info":"Y coordinate of the hit (cm)"},
    {"name":"z",        "type":"F", "info":"Z coordinate of the hit (cm)"},
    {"name":"hx",       "type":"F", "info":"X coordinate of the matched hit (cm)"},
    {"name":"hy",       "type":"F", "info":"Y coordinate of the mathced hit (cm)"},
    {"name":"hz",       "type":"F", "info":"Z coordinate of the matched hit (cm)"},
    {"name":"status",   "type":"S", "info":"hit status"},
```

1 entry per scintillator hit associated with a particle.

The *ith* scintillator entry does not generally correspond to the *ith* REC::particle Entry. Must use **pindex**

https://clasweb.jlab.org/wiki/index.php/CLAS12_DSTs

# Fast Histogramming

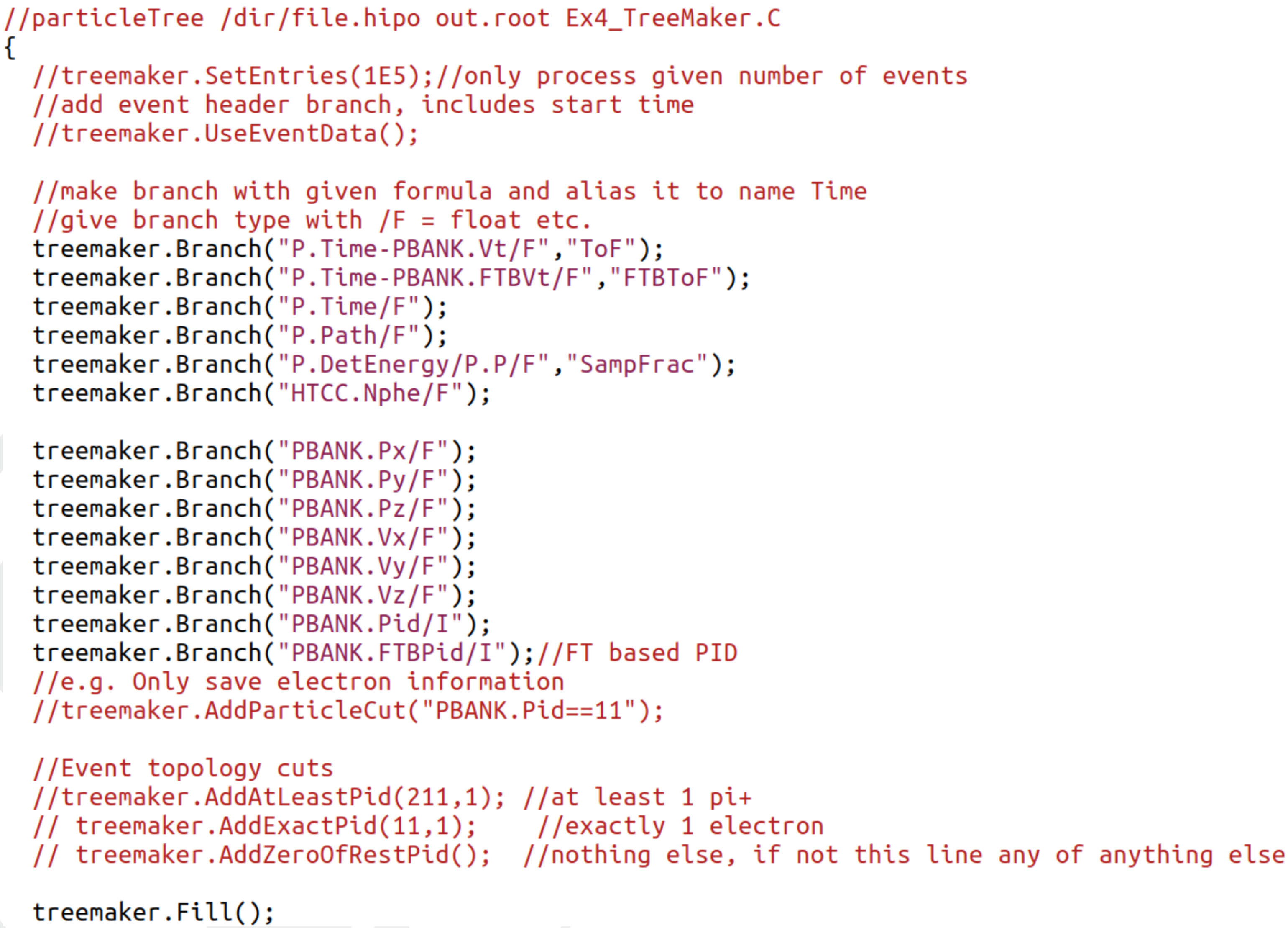


**Draw this** **With this condition**

```
//particleDraw /dir/file.hipo Ex2_HipoDraw.C
{
  hists.Hist1D("P.Theta*TMath::RadToDeg()",180,0,180,"PBANK.Px");
  hists.Hist1D("P.Phi*TMath::RadToDeg()",180,-180,180,"PBANK.Px");
  hists.Hist1D("P.P",100,0.1,12,"PBANK.Px");
  hists.Hist1D("P.Time-EVNT.StartTime",1000,-200,200,"P.Time&&PBANK.Px"); //Note zero suppression

  hists.Hist1D("atan2(sqrt(PBANK.Px*PBANK.Px+PBANK.Py*PBANK.Py),PBANK.Pz)*57.295780",180,0,180,"P.Time&&PBANK.Px");
  hists.Hist1D("atan2(PBANK.Py,PBANK.Px)*TMath::RadToDeg()",180,-180,180,"P.Time&&PBANK.Px");
  hists.Hist1D("sqrt(PBANK.Px*PBANK.Px+PBANK.Py*PBANK.Py+PBANK.Pz*PBANK.Pz)",100,0.1,12);
  hists.Hist1D("FTOF1B.Time-EVNT.StartTime",1000,-200,200,"P.Time&&PBANK.Px&&PBANK.Px");

  hists.Hist1D("P.Theta*TMath::RadToDeg()",180,0,180,"P.Pid==11&&PBANK.Px");
  hists.Hist1D("P.Phi*TMath::RadToDeg()",180,-180,180,"P.Pid==11&&PBANK.Px");
  hists.Hist1D("P.P",100,0.1,12,"P.Pid==11");
  hists.Hist1D("P.Time-EVNT.StartTime",1000,-200,200,"P.Time&&P.Pid==11&&PBANK.Px");

  hists.Hist1D("P.Theta*TMath::RadToDeg()",180,0,180,"P.Pid==2212&&PBANK.Px");
  hists.Hist1D("P.Phi*TMath::RadToDeg()",180,-180,180,"P.Pid==2212&&PBANK.Px");
  hists.Hist1D("P.P",100,0.1,12,"P.Pid==2212&&PBANK.Px");
  hists.Hist1D("P.Time-EVNT.StartTime",1000,-200,200,"P.Time&&P.Pid==2212&&PBANK.Px")->Draw("(4x3)");

  hists.Hist2D("P.Theta*TMath::RadToDeg():P.Phi*TMath::RadToDeg()",180,0,180,180,-180,180,"P.Pid==11");
  hists.Hist2D("P.Theta*TMath::RadToDeg():P.Phi*TMath::RadToDeg()",180,0,180,180,-180,180,"P.Pid==-11");
  hists.Hist2D("P.Theta*TMath::RadToDeg():P.Phi*TMath::RadToDeg()",180,0,180,180,-180,180,"P.Pid==22");
  hists.Hist2D("P.Theta*TMath::RadToDeg():P.Phi*TMath::RadToDeg()",180,0,180,180,-180,180,"P.Pid==2212");
  hists.Hist2D("P.Theta*TMath::RadToDeg():P.Phi*TMath::RadToDeg()",180,0,180,180,-180,180,"P.Pid==211");
  hists.Hist2D("P.Theta*TMath::RadToDeg():P.Phi*TMath::RadToDeg()",180,0,180,180,-180,180,"P.Pid==-211");
  hists.Hist2D("P.Theta*TMath::RadToDeg():P.Phi*TMath::RadToDeg()",180,0,180,180,-180,180,"P.Pid==321");
  hists.Hist2D("P.Theta*TMath::RadToDeg():P.Phi*TMath::RadToDeg()",180,0,180,180,-180,180,"P.Pid==-321")->Draw("(4x2)col1");

  hists.Hist2D("P.P:P.Beta",100,0,10,100,0,2,"P.Beta");
  hists.Hist2D("P.P:P.Beta",100,0,10,100,0,2,"P.Beta&&P.Region==FT");
```

**Fill all simultaneously**
**And plot on 4x3 canvas**

# Create small ROOT trees



```
//particleTree /dir/file.hipo out.root Ex4_TreeMaker.C
{
  //treemaker.SetEntries(1E5);//only process given number of events
  //add event header branch, includes start time
  //treemaker.UseEventData();

  //make branch with given formula and alias it to name Time
  //give branch type with /F = float etc.
  treemaker.Branch("P.Time-PBANK.Vt/F","ToF");
  treemaker.Branch("P.Time-PBANK.FTBVt/F","FTBToF");
  treemaker.Branch("P.Time/F");
  treemaker.Branch("P.Path/F");
  treemaker.Branch("P.DetEnergy/P.P/F","SampFrac");
  treemaker.Branch("HTCC.Nphe/F");

  treemaker.Branch("PBANK.Px/F");
  treemaker.Branch("PBANK.Py/F");
  treemaker.Branch("PBANK.Pz/F");
  treemaker.Branch("PBANK.Vx/F");
  treemaker.Branch("PBANK.Vy/F");
  treemaker.Branch("PBANK.Vz/F");
  treemaker.Branch("PBANK.Pid/I");
  treemaker.Branch("PBANK.FTBPid/I");//FT based PID
  //e.g. Only save electron information
  //treemaker.AddParticleCut("PBANK.Pid==11");

  //Event topology cuts
  //treemaker.AddAtLeastPid(211,1); //at least 1 pi+
  // treemaker.AddExactPid(11,1);    //exactly 1 electron
  // treemaker.AddZeroOfRestPid();  //nothing else, if not this line any of anything else

  treemaker.Fill();
```

# clas12root access to particle data

**Clas12reader::getDetParticles** → C++ vector of **region_particle**s

region_particle points to **REC::Particle** and **REC::Response** data
Responses are already linked using *pindex*

Define a specific region_particle class for FT, FD and CD regions

This allows us to define which detectors may have been hit
and define getTime() etc. differently for each
i.e. use FTOF1B or CTOF

Usage : **p->bank()->getItem();**
Where bank = pbank, sci, cal, che,… i.e REC::Response
**Item = Px, Vz, Pid, Time, Sector,...**

**pbank=REC::Particle**

```
p->pbank()->getPid();
p->sci(FTOF2)->getEnergy();
p->trk(DC)->getSector();
p->che(HTCC)->getNphe();
```

Use specific detector names
FTOF1A, PCAL,...to get its response.
Will be 0 if no hit.

https://github.com/JeffersonLab/clas12root/blob/master/AccesssingBankDataInCpp.txt

# Clas12root particle C++ std::vectors

```
for(auto& p : c12.getDetParticles()) //range based for loop
   if(p->getPid()==11){
      electron.SetXYZM(p->pbank()->getPx(),p->pbank()->getPy(),
                       p->pbank()->getPz,0.0051);

      Double_t eToF = p->getTime()- p->pbank()->getVt();
```

OR (probably more useful, if using EB Pid)

```
auto electrons = c12.getByID(11); //filter DetParticles
if(electrons.empty()==false){
   auto& p = electrons[0];
   electron.SetXYZM(p->pbank()->getPx(),p->pbank()->getPy(),
                    p->pbank()->getPz,0.0051);

   Double_t eToF = p->getTime()- p->pbank()->getVt();
```

Note, what getTime returns depends on which detector region the particle was reconstructed in.
e.g. if FD then FTOF1B or FTOF1A or FTOF2 or PCAL

# Clas12root event information

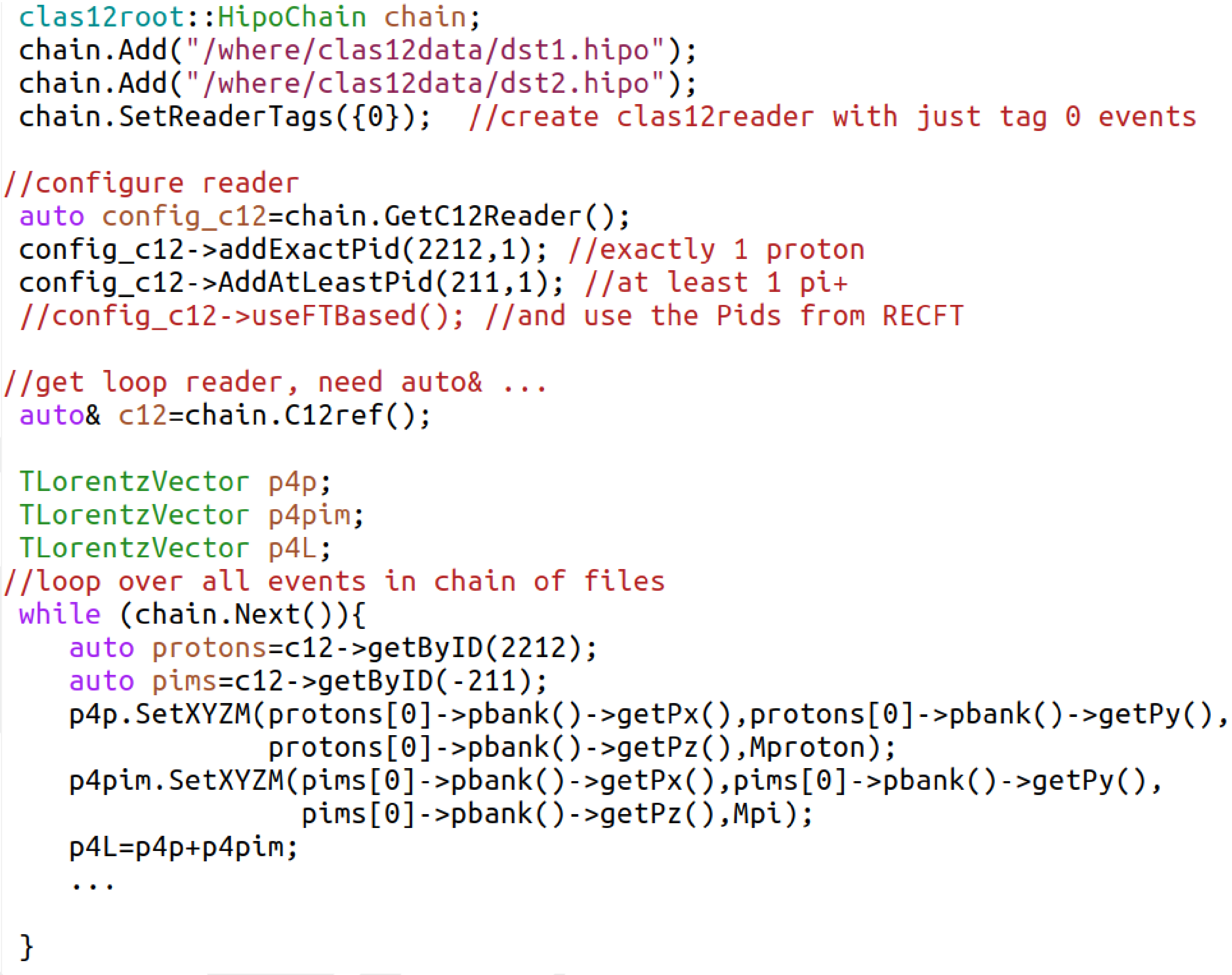
```
clas12reader c12(“file.hipo”);

REC::Event ,
    c12.event()->getStartTime()
   c12.event()->getHelicity()
    …
RUN::config
    c12.runconfig()->getTrigger()  *
    c12.runconfig()->getRun()

MC::Lund
    c12.mcparts()->getPx(i)
    c12.mcparts()->getPid(i)
Or
    c12.mcparts()->setEntry(i);
    c12.mcparts()->getPx();
    c12.mcparts()->getPid();
```

* alternatively for trigger can use

```
            c12.checkTriggerBit(ibit);
```

# Clas12root event loops

## Loop over files

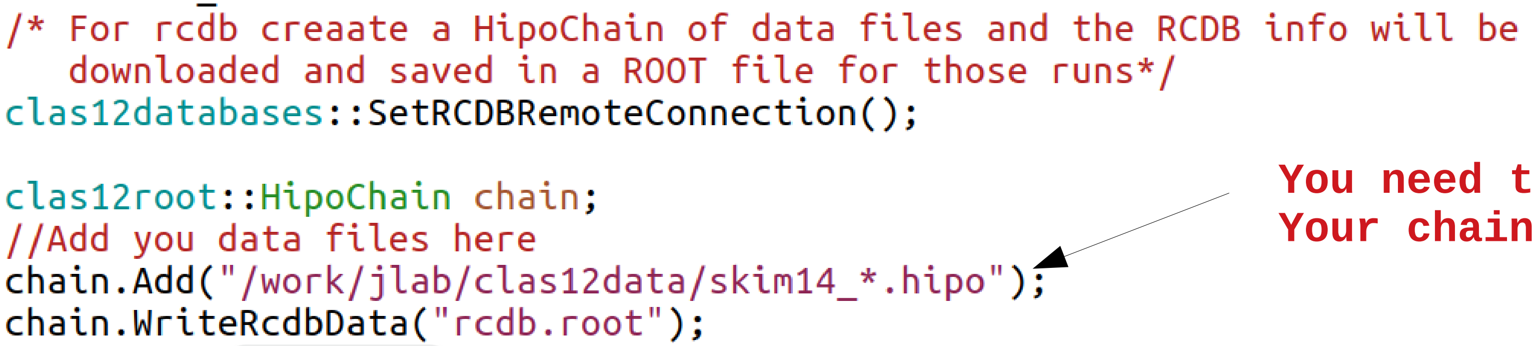


```
 clas12root::HipoChain chain;
 chain.Add("/where/clas12data/dst1.hipo");
 chain.Add("/where/clas12data/dst2.hipo");
 chain.SetReaderTags({0});  //create clas12reader with just tag 0 events

//configure reader
 auto config_c12=chain.GetC12Reader();
 config_c12->addExactPid(2212,1); //exactly 1 proton
 config_c12->AddAtLeastPid(211,1); //at least 1 pi+
 //config_c12->useFTBased(); //and use the Pids from RECFT

//get loop reader, need auto& ...
 auto& c12=chain.C12ref();

 TLorentzVector p4p;
 TLorentzVector p4pim;
 TLorentzVector p4L;
//loop over all events in chain of files
 while (chain.Next()){
    auto protons=c12->getByID(2212);
    auto pims=c12->getByID(-211);
    p4p.SetXYZM(protons[0]->pbank()->getPx(),protons[0]->pbank()->getPy(),
                protons[0]->pbank()->getPz(),Mproton);
    p4pim.SetXYZM(pims[0]->pbank()->getPx(),pims[0]->pbank()->getPy(),
                  pims[0]->pbank()->getPz(),Mpi);
    p4L=p4p+p4pim;
    ...

 }
```

# clas12reader

## Prepare Databases

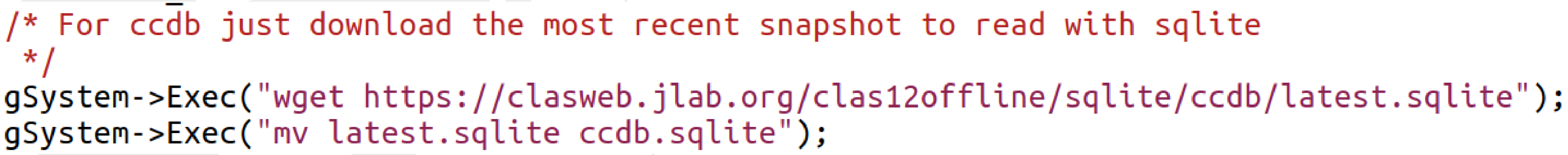
**clas12root PrepareDatabases.C**



```
/* For rcdb creaate a HipoChain of data files and the RCDB info will be
   downloaded and saved in a ROOT file for those runs*/
clas12databases::SetRCDBRemoteConnection();

clas12root::HipoChain chain;
//Add you data files here
chain.Add("/work/jlab/clas12data/skim14_*.hipo");
chain.WriteRcdbData("rcdb.root");
```

**You need to edit**
**Your chain files!!!**

→ **create local root file copy of rcdb for files in your chain**
(because rcdb sqlite would not open sqlite file in C++...)

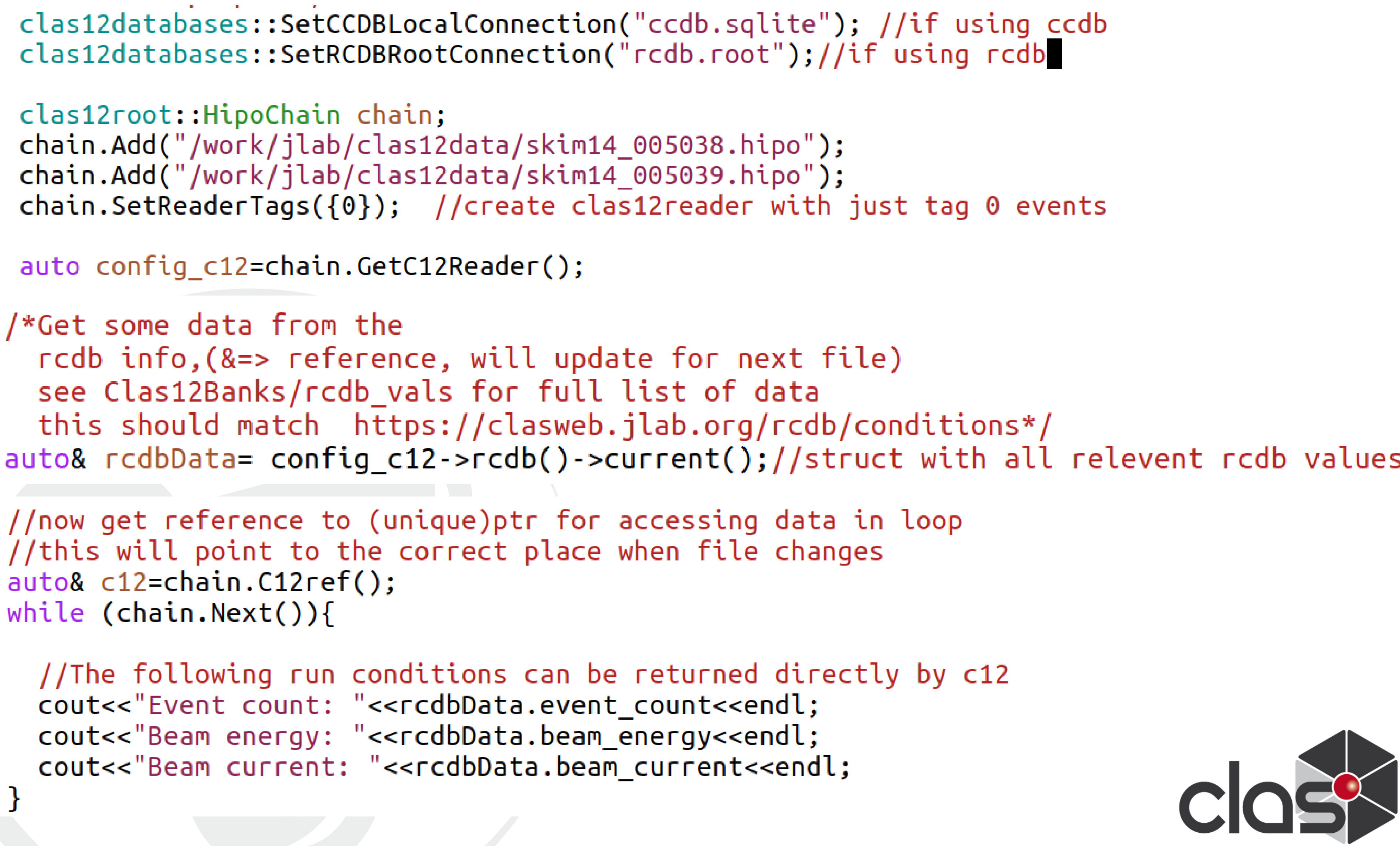


```
/* For ccdb just download the most recent snapshot to read with sqlite
 */
gSystem->Exec("wget https://clasweb.jlab.org/clas12offline/sqlite/ccdb/latest.sqlite");
gSystem->Exec("mv latest.sqlite ccdb.sqlite");
```

Note run # 10 and 11 reserved for simulated data files
Should there be anymore ?

# HipoChain

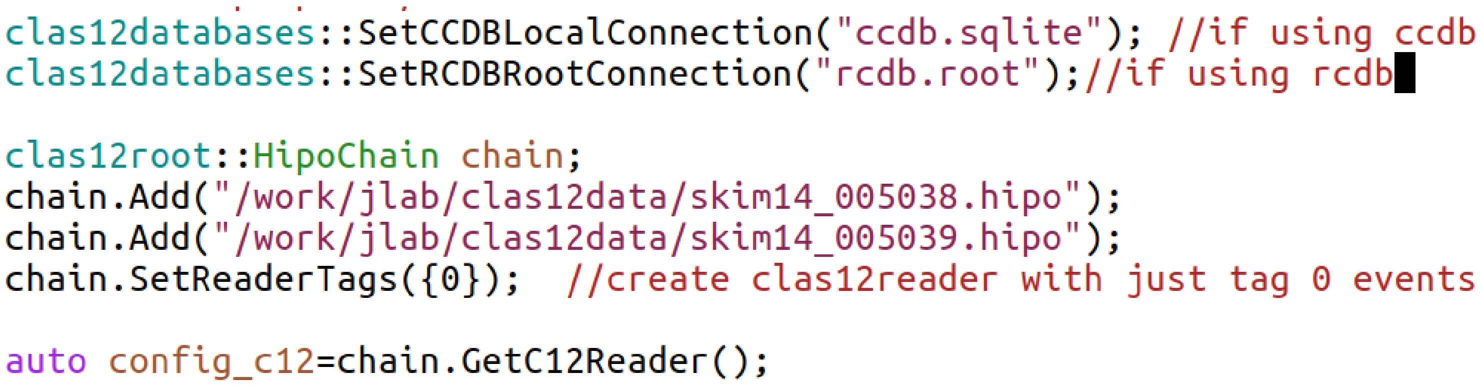
## rcdb access



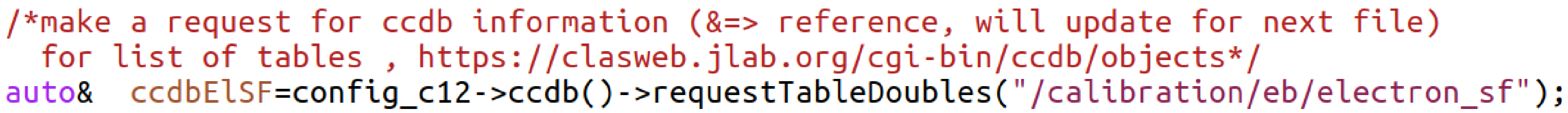
```
 clas12databases::SetCCDBLocalConnection("ccdb.sqlite"); //if using ccdb
 clas12databases::SetRCDBRootConnection("rcdb.root");//if using rcdb

 clas12root::HipoChain chain;
 chain.Add("/work/jlab/clas12data/skim14_005038.hipo");
 chain.Add("/work/jlab/clas12data/skim14_005039.hipo");
 chain.SetReaderTags({0});  //create clas12reader with just tag 0 events

 auto config_c12=chain.GetC12Reader();

/*Get some data from the
  rcdb info,(&=> reference, will update for next file)
  see Clas12Banks/rcdb_vals for full list of data
  this should match  https://clasweb.jlab.org/rcdb/conditions*/
auto& rcdbData= config_c12->rcdb()->current();//struct with all relevent rcdb values

//now get reference to (unique)ptr for accessing data in loop
//this will point to the correct place when file changes
auto& c12=chain.C12ref();
while (chain.Next()){

  //The following run conditions can be returned directly by c12
  cout<<"Event count: "<<rcdbData.event_count<<endl;
  cout<<"Beam energy: "<<rcdbData.beam_energy<<endl;
  cout<<"Beam current: "<<rcdbData.beam_current<<endl;
}
```

# clas12reader

## ccdb access

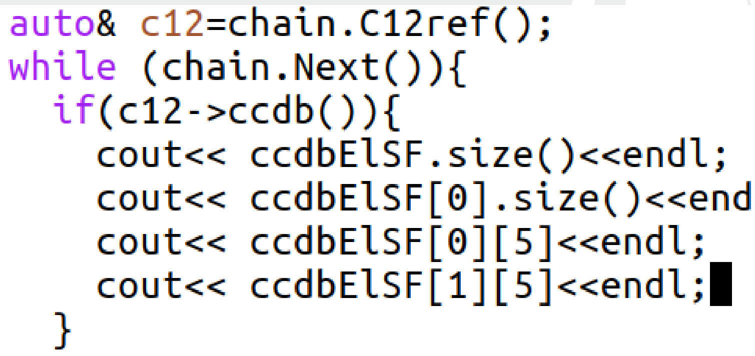


```
clas12databases::SetCCDBLocalConnection("ccdb.sqlite"); //if using ccdb
clas12databases::SetRCDBRootConnection("rcdb.root");//if using rcdb

clas12root::HipoChain chain;
chain.Add("/work/jlab/clas12data/skim14_005038.hipo");
chain.Add("/work/jlab/clas12data/skim14_005039.hipo");
chain.SetReaderTags({0});  //create clas12reader with just tag 0 events

auto config_c12=chain.GetC12Reader();
```



```
/*make a request for ccdb information (&=> reference, will update for next file)
  for list of tables , https://clasweb.jlab.org/cgi-bin/ccdb/objects*/
auto&  ccdbElSF=config_c12->ccdb()->requestTableDoubles("/calibration/eb/electron_sf");
```

Returns a vector<vector<double>>;

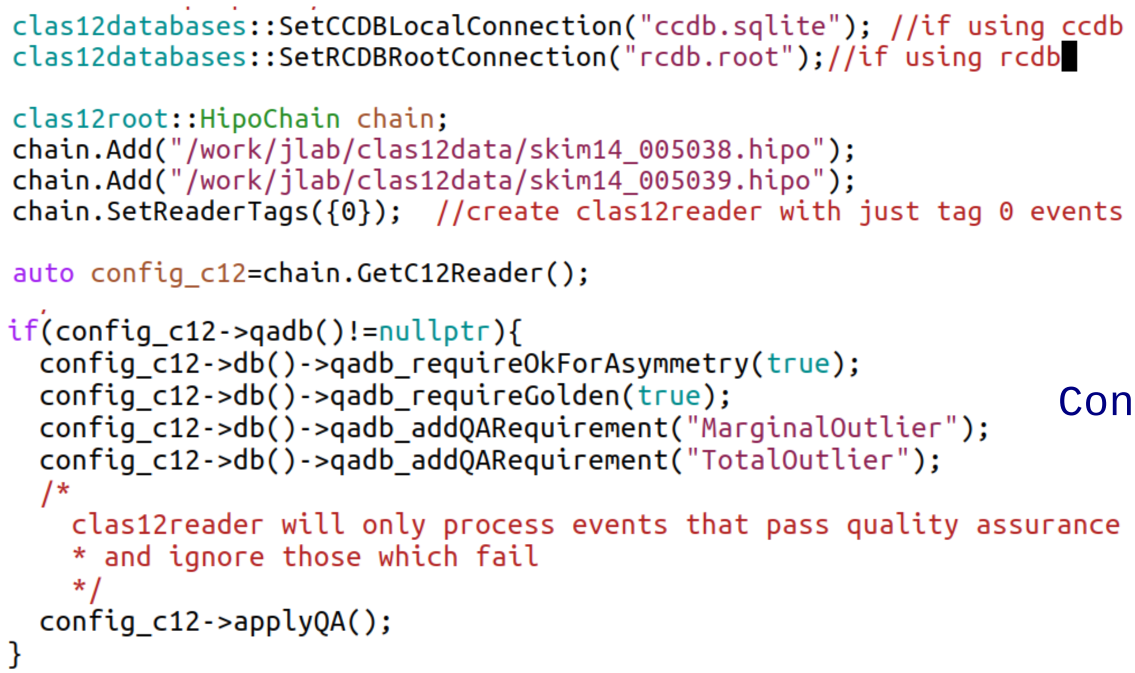


```
auto& c12=chain.C12ref();
while (chain.Next()){
  if(c12->ccdb()){
    cout<< ccdbElSF.size()<<endl;
    cout<< ccdbElSF[0].size()<<endl;
    cout<< ccdbElSF[0][5]<<endl;
    cout<< ccdbElSF[1][5]<<endl;
  }
```

Table values will update
When file changes

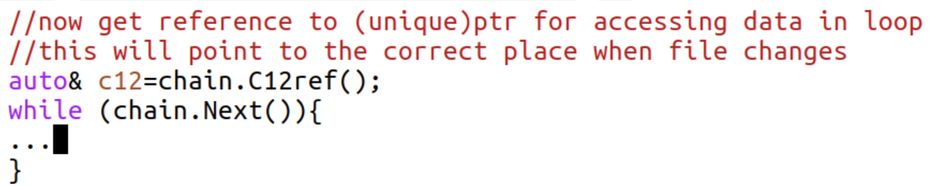
# clas12reader

# qcdb



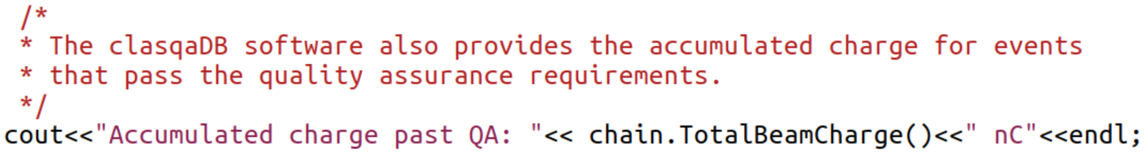
```
clas12databases::SetCCDBLocalConnection("ccdb.sqlite"); //if using ccdb
clas12databases::SetRCDBRootConnection("rcdb.root");//if using rcdb

clas12root::HipoChain chain;
chain.Add("/work/jlab/clas12data/skim14_005038.hipo");
chain.Add("/work/jlab/clas12data/skim14_005039.hipo");
chain.SetReaderTags({0});  //create clas12reader with just tag 0 events

auto config_c12=chain.GetC12Reader();

if(config_c12->qadb()!=nullptr){
  config_c12->db()->qadb_requireOkForAsymmetry(true);
  config_c12->db()->qadb_requireGolden(true);
  config_c12->db()->qadb_addQARequirement("MarginalOutlier");
  config_c12->db()->qadb_addQARequirement("TotalOutlier");
  /*
    clas12reader will only process events that pass quality assurance
    * and ignore those which fail
    */
  config_c12->applyQA();
}
```

Configure qadb requirements



```
//now get reference to (unique)ptr for accessing data in loop
//this will point to the correct place when file changes
auto& c12=chain.C12ref();
while (chain.Next()){
...
}
```

Only events passing qa processed



```
 /*
 * The clasqaDB software also provides the accumulated charge for events
 * that pass the quality assurance requirements.
 */
cout<<"Accumulated charge past QA: "<< chain.TotalBeamCharge()<<" nC"<<endl;
```

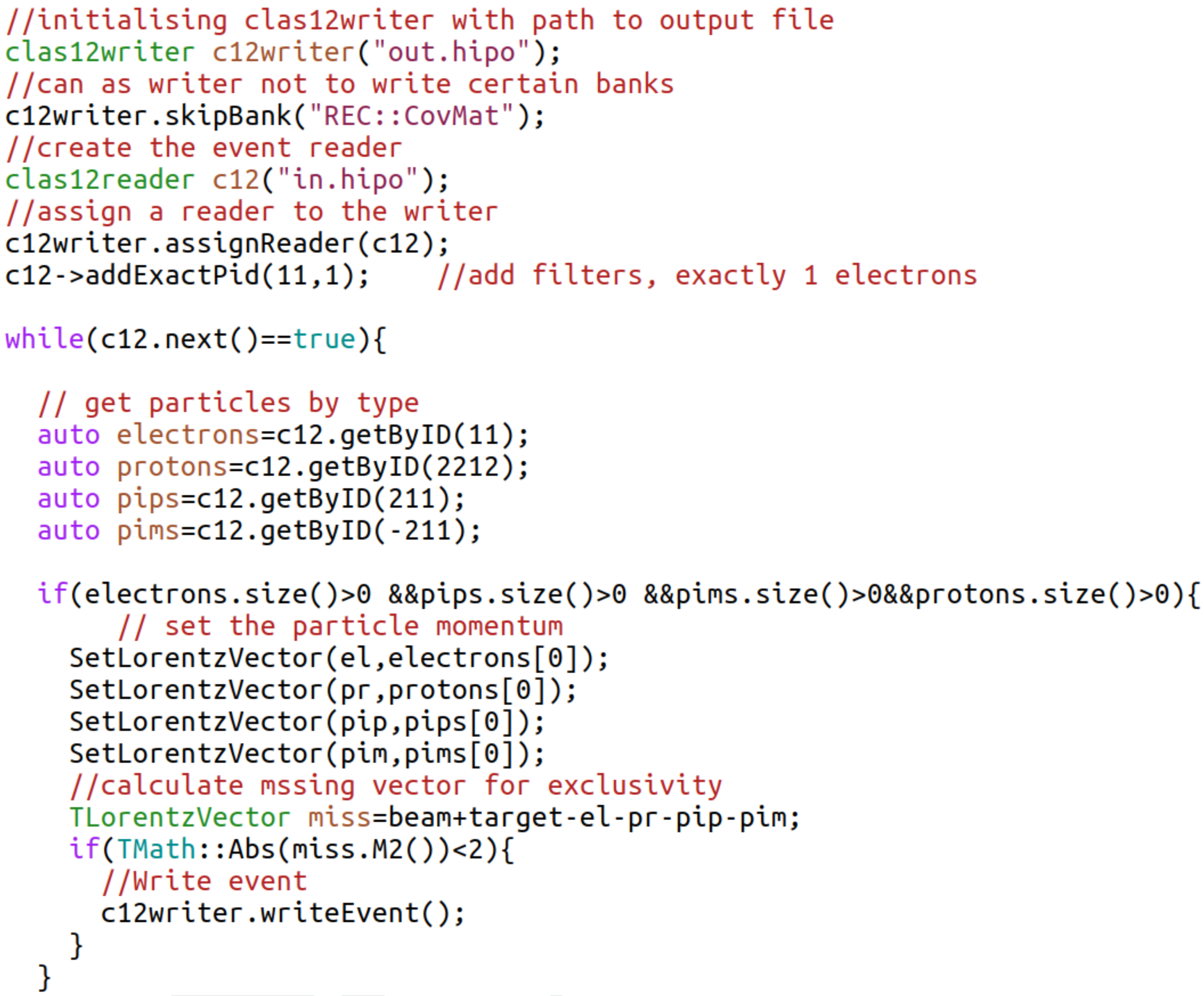
Total charge for files that pass your qadb conditions

# Clas12reader
# Additional banks

```
clas12reader c12{“file.hipo”};

//get track based hits id and layers
//Add extra bank for reading and get its ID

auto idx_TRCKHits= c12.addBank("TimeBasedTrkg::TBHits");

//Add an item in the bank for reading and get its ID

auto iTrckId =  c12.getBankOrder(idx_TRCKHits,"id");
auto iTrckLayer =  c12.getBankOrder(idx_TRCKHits,"layer");

while(c12.next()==true){

auto Nhits=c12.getBank(idx_TRCKHits)->getRows();
//loop over time based tracks for this event
for(auto itr=0;itr<Nhits;itr++){

    auto id=c12.getBank(idx_TRCKHits)->getInt(iTrckId,itr);
    auto layer=c12.getBank(idx_TRCKHits)->getInt(iTrckLayer,itr);
```

# clas12writer

## Tool to produce small files with selected events and banks

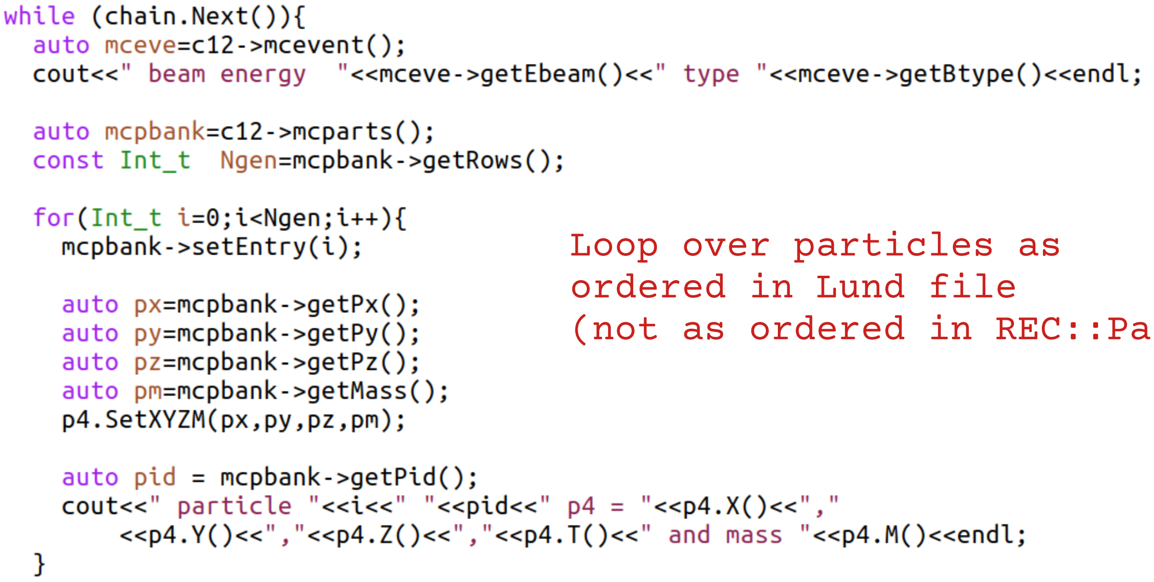


```
//initialising clas12writer with path to output file
clas12writer c12writer("out.hipo");
//can as writer not to write certain banks
c12writer.skipBank("REC::CovMat");
//create the event reader
clas12reader c12("in.hipo");
//assign a reader to the writer
c12writer.assignReader(c12);
c12->addExactPid(11,1);    //add filters, exactly 1 electrons

while(c12.next()==true){

  // get particles by type
  auto electrons=c12.getByID(11);
  auto protons=c12.getByID(2212);
  auto pips=c12.getByID(211);
  auto pims=c12.getByID(-211);

  if(electrons.size()>0 &&pips.size()>0 &&pims.size()>0&&protons.size()>0){
       // set the particle momentum
    SetLorentzVector(el,electrons[0]);
    SetLorentzVector(pr,protons[0]);
    SetLorentzVector(pip,pips[0]);
    SetLorentzVector(pim,pims[0]);
    //calculate mssing vector for exclusivity
    TLorentzVector miss=beam+target-el-pr-pip-pim;
    if(TMath::Abs(miss.M2())<2){
      //Write event
      c12writer.writeEvent();
    }
  }
```

# clas12reader

# Simulated data

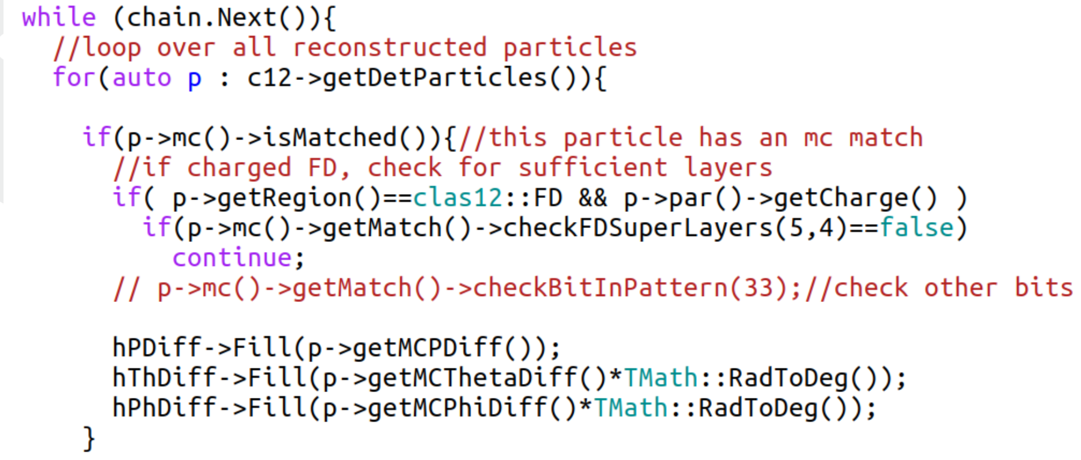
Use mcevent()
And mcparts()



```
while (chain.Next()){
  auto mceve=c12->mcevent();
  cout<<" beam energy  "<<mceve->getEbeam()<<" type "<<mceve->getBtype()<<endl;

  auto mcpbank=c12->mcparts();
  const Int_t  Ngen=mcpbank->getRows();

  for(Int_t i=0;i<Ngen;i++){
    mcpbank->setEntry(i);

    auto px=mcpbank->getPx();
    auto py=mcpbank->getPy();
    auto pz=mcpbank->getPz();
    auto pm=mcpbank->getMass();
    p4.SetXYZM(px,py,pz,pm);

    auto pid = mcpbank->getPid();
    cout<<" particle "<<i<<" "<<pid<<" p4 = "<<p4.X()<<","
        <<p4.Y()<<","<<p4.Z()<<","<<p4.T()<<" and mass "<<p4.M()<<endl;
  }
```

Loop over particles as
ordered in Lund file
(not as ordered in REC::Particle!)

## For when truth matching included in simulated DSTs



```
while (chain.Next()){
  //loop over all reconstructed particles
  for(auto p : c12->getDetParticles()){

    if(p->mc()->isMatched()){//this particle has an mc match
      //if charged FD, check for sufficient layers
      if( p->getRegion()==clas12::FD && p->par()->getCharge() )
        if(p->mc()->getMatch()->checkFDSuperLayers(5,4)==false)
          continue;
      // p->mc()->getMatch()->checkBitInPattern(33);//check other bits

      hPDiff->Fill(p->getMCPDiff());
      hThDiff->Fill(p->getMCThetaDiff()*TMath::RadToDeg());
      hPhDiff->Fill(p->getMCPhiDiff()*TMath::RadToDeg());
    }
```

# clas12reader

## Summary

Clas12root provides access to all data required for analysing Clas12 reactions

All particle and detector information from HiPO DSTs
MC truth information from MC::Lund banks
All database information from rc,cc and qa Dbs

It can be run in ROOT scripts, included in compiled C++ code
Or used in Jupyter notebooks via pyROOT.

Fast drawing and skimming routines available (see docs)

DST filtering from clas12writer