EP578 Computing for Physicists

Topic 12

Advanced Trees

Department of
Engineering Physics

University of Gaziantep

Course web page
www.gantep.edu.tr/~bingul/ep578

Jan 2012

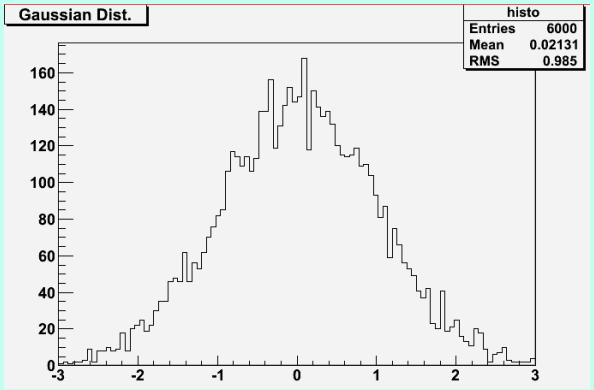
Introduction

We will consider some advanced applications of root.

- ROOT in C++ programs
- How to add vector objects in trees
- Using MakeClass()

Using ROOT Classes in C++

```
// ROOT in C
#include <iostream>
#include <cmath>
using namespace std;

// ROOT
#include "TROOT.h"
#include "TH1F.h"
#include "TRandom3.h"
#include "TApplication.h"

int main(int argc, char **argv)
{
  TApplication *rootApp = new TApplication("example",&argc, argv);
  TH1F *h = new TH1F("histo","Gaussian Dist.",100,-3,3);
  TRandom3 *r = new TRandom3();

  for(int i=0; i<6000; i++){
   h->Fill( r->Gaus() );
  }

  h->Draw();
  rootApp->Run();

} // end of main
```



```
#-------------------------------------------------------------
# Use this script if you are compiling c++ with root.
#!/bin/sh

if [ "$1" = "" ]; then
  echo "Usage: compile "
  echo "       where .cc exists"
  exit
fi

src=$1.cc
if [ ! -r $src ]; then
  echo "Can't find source file $src"
  exit
fi

exe=${src%.*}.exe
echo "Compiling $src with root"

g++ $src -o $exe -O2 -ansi -W -Wall -Wshadow      \
    `$ROOTSYS/bin/root-config --cflags`           \
    `$ROOTSYS/bin/root-config --glibs` -lMinuit   \
    -I$ROOTSYS/include                            \
    -pthread

if [ $? -eq 1 ]; then
  echo "COMPILE FAILED!"
else
  echo "DONE -> $exe"
fi
```

Compiling and running the file main.cc:

```
$ compile-root main
$ ./main.exe
```

Putting Vectors in Trees

We can also put vector objects to a tree.

```
void tree2w(){
    gROOT->Reset();
// vector pointers
    std::vector<float> *r = new vector<float>;
    std::vector<float> *h = new vector<float>;
    std::vector<int>   *a = new vector<int>;
    std::vector<int>   *b = new vector<int>;
// data files and trees
    TFile *file = new TFile("Geometry.root","recreate");
    TTree *t1   = new TTree("Cylinder", "Cylinder data");
    TTree *t2   = new TTree("Rectangle","Rectangle data");
// set branches
    t1->Branch("r", &r);
    t1->Branch("h", &h);
    t2->Branch("a", &a);
    t2->Branch("b", &b);

// continue in the next page ...
```

```
// Fill the trees
    for(int i=1; i<=1000; i++){
      for(int j=0; j<10; j++){
         r->push_back(2+i);
         h->push_back(i*i);
      }
      for(int j=0; j<5; j++){
         a->push_back(i);
         b->push_back(i+j);
      }
      t1->Fill();
      t2->Fill();
      r->clear(); h->clear();
      a->clear(); b->clear();
    }
    t1->Write();
    t2->Write();
    file->Close();
    cout << "Done." << endl;
}
```

Advanced Trees

We have some files generated by Pythia 8.153 Event Generator at:

http://www1.gantep.edu.tr/~bingul/hep/student

```
-rw-r--r-- 1 bingul users 1442795 2012-01-12 07:19 pythia_minbias_2k1_14TeV.root
-rw-r--r-- 1 bingul users 1481987 2012-01-12 07:23 pythia_minbias_2k2_14TeV.root
-rw-r--r-- 1 bingul users 1442795 2012-01-12 07:24 pythia_minbias_2k3_14TeV.root
-rw-r--r-- 1 bingul users 1442795 2012-01-12 07:25 pythia_minbias_2k4_14TeV.root
-rw-r--r-- 1 bingul users 1507422 2012-01-12 07:27 pythia_minbias_2k5_14TeV.root
```

- The files contain visible final state particles from minimum bias events at 14 TeV p-p collisions.
- All files have same content (~2000 minbias events)!

The leafs are:

Name	Type	Description
m_px	float	x component of the particle momentum in GeV/c
m_py	float	y component of the particle momentum in GeV/c
m_pz	float	z component of the particle momentum in GeV/c
m_xProd	float	x coordinate of the production point in mm
m_yProd	float	y coordinate of the production point in mm
m_zProd	float	z coordinate of the production point in mm
m_id	int	particle's id number
m_motherid1	int	particle's first mother id number
m_motherid2	int	particle's second mother id number
m_mother1	int	particle's first mother barcode (track) number
m_mother2	int	particle's second mother barcode (track) number

Look at the following pages for particle codes:

http://www.physics.ox.ac.uk/CDF/Mphys/old/notes/pythia_codeListing.html

http://cepa.fnal.gov/psm/simulation/mcgen/lund/pythia_manual/pythia6.3/pythia6301/node34.html

Using MakeClass

```
$ root pythia_minbias_2k1_14TeV.root
```

To use default name:

```
root[0] OfflineTracks->MakeClass()
Info in <TTreePlayer::MakeClass>: Files:
OfflineTracks.h and OfflineTracks.C generated from
TTree: OfflineTracks
```

To use you your own name:

```
root[0] OfflineTracks->MakeClass("MyClass")
Info in <TTreePlayer::MakeClass>:
Files: MyClass.h and MyClass.C generated from
TTree: OfflineTracks
```

Note that

`MyClass.h`

contains the class definition of "MyClass"

`MyClass.C`

contains the class implementation of "MyClass"

Loading and Using MyClass.C

Load the macro and create a MyClass object:

```
root [0].L MyClass.C
root [1] MyClass *m = new MyClass ();
```

Use **`MyClass::GetEntry()`**

```
root [2] m->GetEntry(1)  // get first entry (line)
(Int_t)44
root [3] m->m_px
(Float_t)6.5561002492904631e-01
root [4] m->m_py
(Float_t)1.4197568934326172e+00
root [5] m->m_pz
(Float_t)(-6.1720693111419677e-01)
```

Modify the MyClass.C as follows:

```
void MyClass::Loop()
{
   if (fChain == 0) return;

   Long64_t nentries = fChain->GetEntriesFast();

   Long64_t nbytes = 0, nb = 0;
   for (Long64_t jentry=0; jentry<nentries;jentry++) {
      nb = fChain->GetEntry(jentry);   nbytes += nb;
      cout << nbytes << ": ";
      cout << m_px << " " << m_py << " " << m_pz << endl;
   }
}
```

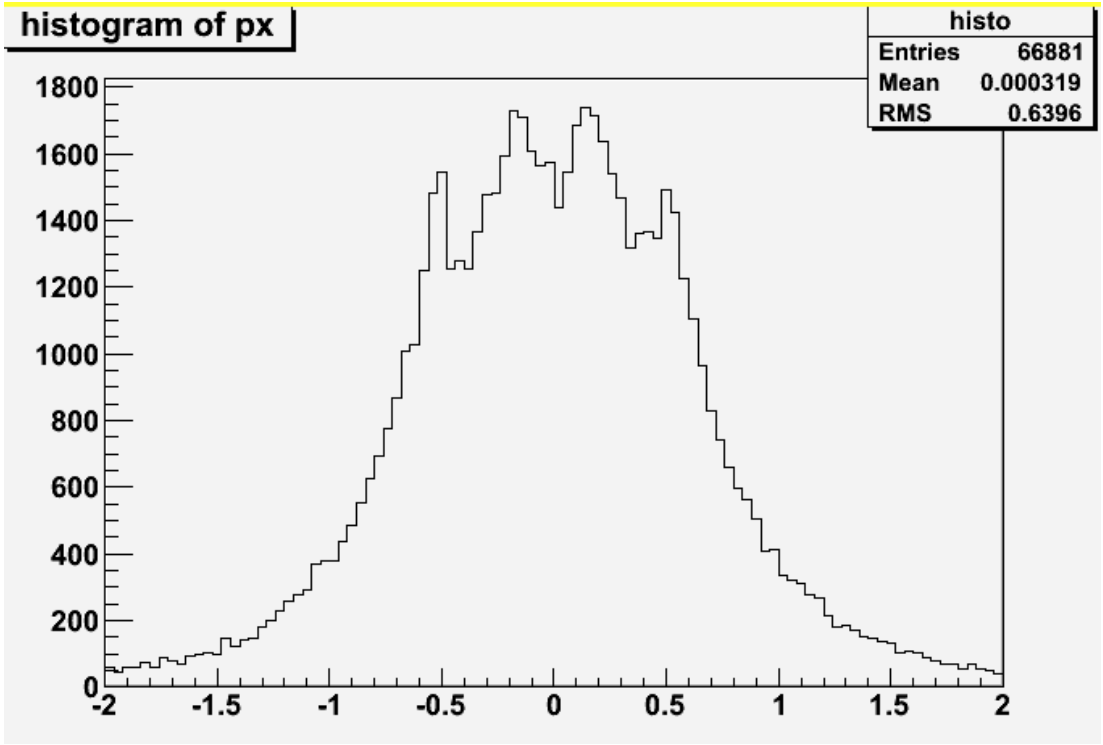
```
root [0].L MyClass.C
root [1] MyClass *m = new MyClass();
root [2] m->Loop()
44: -0.0713867 -0.618236 -0.113392
88:  0.65561 1.41976 -0.617207
132: 0.725295 -0.100082 0.015592
176: 0.146356 -1.34271 0.384952
```

Expanding Loop()

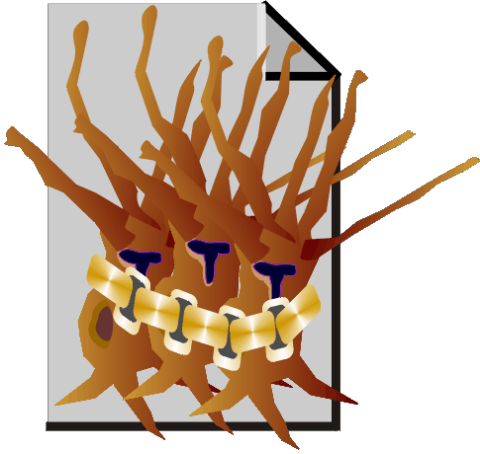
```
void MyClass::Loop()
{
   if (fChain == 0) return;
   TH1F *h = new TH1F("histo","histogram of px",100,-2,2);

   Long64_t nentries = fChain->GetEntriesFast();

   Long64_t nbytes = 0, nb = 0;
   for (Long64_t jentry=0; jentry<nentries;jentry++) {
      nb = fChain->GetEntry(jentry);   nbytes += nb;
      //cout << nbytes << endl;
      //cout << m_px << " " << m_py << " " << m_pz << endl;
      if(m_px !=0) h->Fill(m_px);
   }
   h->Draw();
}
```

Load the macro and create a MyClass object:

```
root [0].L MyClass.C
root [1] MyClass *m = new MyClass ();
root [2] m->Loop()
```



Chains



The aim is to perform an analysis using multiple ROOT files.

All files are of the same structure and have the same tree.

```
pythia_minbias_2k1_14TeV.root
pythia_minbias_2k2_14TeV.root
pythia_minbias_2k3_14TeV.root
pythia_minbias_2k4_14TeV.root
pythia_minbias_2k5_14TeV.root
```

```
TChain *ch = new TChain("OfflineTracks");
ch->Add("pythia_minbias_2k*_14TeV.root");
```

Summary & Full Analysis in C++ Code

1. Open a root file:

```
$ root pythia_minbias_2k1_14TeV.root
```

2. Make a class

```
root[0] OfflineTracks->MakeClass("MyClass")
root[1] .q
```

3. Generate a main (driver) program under the same folder that you work on:

```
// http://www1.gantep.edu.tr/~bingul/hep/student/main.cc
#include <TApplication.h>
#include <TTree.h>
#include <TChain.h>
#include <TROOT.h>
#include <iostream>
#include "MyClass.h"
#include "MyClass.C"
using namespace std;

int main(int argc, char **argv){
  TApplication *rootApp = new TApplication("example",&argc, argv);
// Get chain
  TChain *ch = new TChain("OfflineTracks");
  ch->Add("pythia_minbias_2k*_14TeV.root");
  MyClass *par = new MyClass(ch);
// Get the number of events (same for all trees)
  Long64_t nline = par->fChain->GetEntries();

  for (Long64_t i=0; i<nline; i++){ // Main loop
     par->GetEntry(i); // get ith entry
     if(par->m_px==0 && par->m_py==0 && par->m_pz==0){// eventa are sepataed by 0 0 0 0 0 0 0 0 0 0
       cout << "--- End of the event ------------------"<<endl;
       continue;
     }

     cout << par->m_id <<" "
          << par->m_px <<" "<< par->m_py <<" "<< par->m_pz <<" | "
          << par->m_motherid1 <<" "<<" "<< par->m_mother1 << endl;

  } // end of main loop

  cout << "End." << endl;
  rootApp->Run();
  return 0;
}
```

4. Compile and run the program:

```
$ compile-root main
$ ./main
```

```
211 -0.0713867 -0.618236 -0.113392 | 2  591
211 0.65561 1.41976 -0.617207 | 1  627
2212 0.725295 -0.100082 0.015592 | 1  627
-211 0.146356 -1.34271 0.384952 | 1  627
-211 -1.03415 0.89702 0.355303 | 1  684
211 0.485675 0.169107 2.94966 | 1  684
. . .
22 0.389638 0.460049 1.84146 | 111  978
22 0.187086 0.18691 -1.10462 | 111  955
--- End of the event -----------------
-211 0.778032 -0.53409 -8.50529 | 2  130
211 -0.0391108 0.779651 -0.758457 | 2  130
321 0.190398 0.780868 2.90514 | 313  195
-211 0.250862 -0.438566 1.3314 | -213  197
211 -0.58004 -0.314647 -1.46089 | 213  199
22 0.0386902 -0.209163 1.94537 | 111  228
22 0.420434 0.518341 -1.58915 | 111  238
22 0.234559 -0.19269 -2.35157 | 111  240
--- End of the event -----------------
321 -0.64306 0.213484 3.52188 | 1  240
. . .
```

Homeworks

Use all files **`pythia_minbias_2k*_14TeV.root`**

1. On the same canvas draw
   a) the pseudo-rapidity distribution of all charged tracks
   b) the transverse momentum distribution of all charged tracks
   c) the pseudo-rapidity distribution of all photons
   d) the energy distributions of all photons

2. On the same canvas draw
   a) the energy distribution of all charged pions
   b) the energy distribution of all neural pions
   c) the energy distribution of all photons originating from neutral pions
   d) the energy distribution of all photons not originating from neutral pions

3. Consider the decay: $\rho^0 \rightarrow \pi^+ \pi^-$. On the same canvas draw

a) the momentum distribution of neutral rho mesons decaying two pions

b) the invariant mass distribution of neutral rho mesons decaying two pions

c) the transverse momentum distribution of charged pions originating from neutral rho mesons

d) the opening angle distribution between charged pions originating from a neutral rho meson

4. Consider the decay: $Ks^0 \rightarrow \pi^+ \pi^-$. On the same canvas draw

a) the energy distribution of neutral kaons decaying two charged pions

b) the momentum distribution of charged pions originating from neutral kaons

c) the distribution of the distance between production point (xProd, yProd, zProd) of Ks^0 and interaction point (0,0,0).

d) the distribution of the lab-frame lifetime of Ks^0 particles