(C++) ROOT & ALICE Data Analysis (++ More ??!!)

Indranil Das

indranil.das@cern.ch

Outline : Life cycle of EHEP PhD student

1. C++ language
2. ROOT : HEP analysis tool
3. AliRoot : ALICE Analysis Software
4. Various

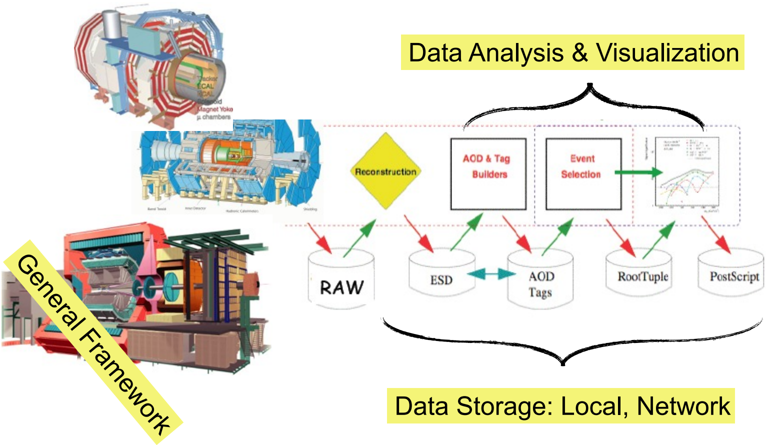
ROOT in a nutshell (GRIDKA-2013)

- Framework for large scale data handling
- Provides, among others,
  - an efficient data storage, access and query system (PetaBytes)
  - advanced statistical analysis: histogramming, fitting, minimization and multi-variate analysis algorithms
  - scientific visualization: 2D and 3D graphics, Postscript, PDF, LateX
  - geometrical modeler
  - PROOF parallel query engine
- An Open Source Project

ROOT in a nutshell (GRIDKA-2013)

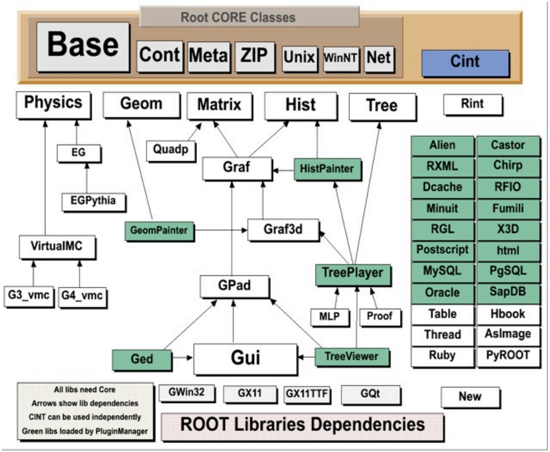
- The analysis of data coming from LHC experiments (and also other experiments) requires a powerful and general toolkit
  - Visualisation
  - Statistical studies
  - Data reduction
  - Multivariate techniques
- A scalable and reliable persistency method is needed to write the data on disks and tapes.

ROOT in a nutshell (GRIDKA-2013)



ROOT in a nutshell (GRIDKA-2013)

- Overview of ROOT libraries and their dependencies
  - 1,700,000 lines of code.
  - More than 100 shared libraries
  - Fully cross-platform: Unix/ Linux, MacOS and Windows.
  - More than 10000 downloads every month



ROOT basics

- ROOT : An Object-Oriented Data Analysis Framework
- Download : https://root.cern.ch/downloading-root
- Installation : Installation from source code (click here)
- Forum : ROOT discussion (click here)

Add alias in your "$HOME/.bashrc"

```
alias set_school_root = 'source
$YOUR_ROOT_INSTALLATION_PATH/bin/thisroot.sh'
```

ROOT session

- Start root session from terminal and quit from session

```
[user@Rjn-Inlap]$ root -l
root [0] .q
```

- Execute a root macro “code.C” interpreter mode

```
[user@Rjn-Inlap]$ root -l code.C
```

- Compile, run and send the stdout and stderr of “code.C” to output.log

```
[user@Rjn-Inlap]$ root -l -n -b -q code.C+ > output.log 2>&1
```

ROOT session

If some libraries are to be loaded or header file path to appended at the starting of root session, it can be set by creating rootlogon.C

```
{
TString includePath = "-I$ALICE_ROOT/include ";
includePath += "-I$ALICE_ROOT/RAW ";
gSystem->SetIncludePath(includePath.Data());

gSystem->Load("libAliHLTMUON");

cout << "loading user specific header file path and libraries" << endl;
}
```

Caution

Note that the path of library libAliHLTMUON.so must be included in your LD_LIBRARY_PATH environment before you start the "root" session, otherwise the library loading will not work. However, this will not stop you to start a "root" session.

CINT/Cling

- CINT/Cling : ROOT's C++ Interpreter
- Commands starting with "."

```
root [0] .? // to list all the interpreter commands
root [1] .L Code.C // to load macro
root [2] .x Code.C // to execute macro
root [3] .x Code.C+ // to compile and execute macro
```

- Commands starting with ".!" will run shell commands

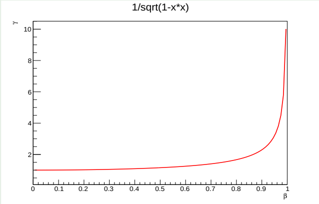
```
root [0] .! hostname
Rjn-Inlap
root [1] .! whoami
user
```

- TAB completion feature is also an important feature

Function

- ROOT user can define and plot 1D, 2D or 3D

```
[user@Rjn-Inlap]$ root -l
root [0] TF1 *fn = new TF1(”fn”,”1/sqrt(1-x*x)”,0,1)
root [1] fn->GetXaxis()->SetTitle(”#beta”)
root [2] fn->GetYaxis()->SetTitle(”#gamma”)
root [3] fn->Draw()
Info in <TCanvas::MakeDefCanvas>: created default TCanvas
with name c1
root [4] gROOT->GetListOfCanvases()->At(0)->
SaveAs(”beta_vs_gamma.pdf”)
Info in <TCanvas::Print>: pdf file beta_vs_gamma.pdf has been
created
```



In daily life 1D function is mostly used with multiple parameters, followed by 2D functions.

Function

The function can be declared inside a macro Code.C

```
#include <TF1.h>
#include <TAxis.h>
#include <TCanvas.h>

Double_t Lorentz_factor(Double_t *x, Double_t *par)
{
  return 1.0/sqrt(1-x[0]*x[0]) ;
}

Bool_t Code()
{

  TF1 *fn = new TF1("fn",Lorentz_factor,0,1,0);
  fn->GetXaxis()->SetTitle("#beta");
  fn->GetYaxis()->SetTitle("#gamma");

  TCanvas *c1 = new TCanvas("c1","c1");
  fn->Draw();
  c1->Update();
  c1->SaveAs("beta_gamma.pdf");

  return kTRUE;
}
```

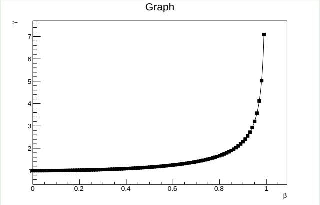
and then executed as below to plot it in canvas

```
[user@Rjn-Inlap]$ root -l code.C++
```

Graph

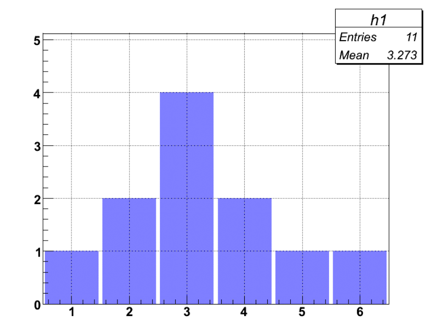
Useful methods TGraph, TGraphErrors, TGraphAsymmErrors, and TMultiGraph

```
#include <TF1.h>
#include <TAxis.h>
#include <TCanvas.h>
#include <TGraph.h>

Double_t Lorentz_factor(Double_t *x, Double_t *par)
{
  return 1.0/sqrt(1-x[0]*x[0]) ;
}

Bool_t CodeGraph()
{
  TF1 *fn = new TF1("fn",Lorentz_factor,0,1,0);

  Int_t n = 100;
  Double_t x[n], y[n];
  for (Int_t i=0; i<n; i++) {
    x[i] = i*0.01;
    y[i] = fn->Eval(x[i]);
  }
  TGraph *gr = new TGraph (n, x, y);
  gr->SetMarkerStyle(kFullSquare);
  gr->GetXaxis()->SetTitle("#beta");
  gr->GetYaxis()->SetTitle("#gamma");

  TCanvas *c1 = new TCanvas("c1","c1");
  gr->Draw("ACP");
  c1->Update();
  c1->SaveAs("beta_gamma.pdf");

  return kTRUE;
}
```



Histogram (CSC-2011)

- Histogram is just occurrence counting, i.e. how often they appear
- Example: {1,3,2,6,2,3,4,3,4,3,5}



Histogram (CSC-2011)

- **How is a Real Histogram Made?**
  Lets consider the age distribution of the CSC participants in 2008:

Ages
26.93
23.03
29.66
33.74
32.19
31.40
24.88
27.56
26.99
26.65

Age	Number
20-22	1
22-24	5
24-26	14
26-28	14
28-30	10
30-32	5
32-34	5
34-36	1
36-38	1
38-40	1

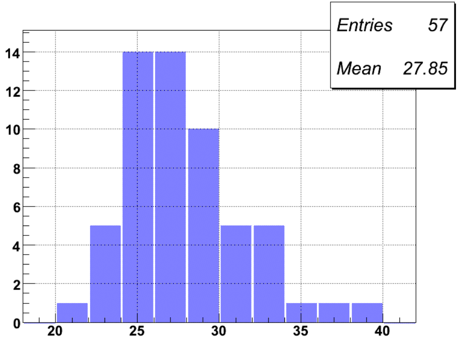
Binning:

Grouping ages of participants in several categories (bins)

Histogram (CSC-2011)

Table of Ages (binned)

Age	Number
20-22	1
22-24	5
24-26	14
26-28	14
28-30	10
30-32	5
32-34	5
34-36	1
36-38	1
38-40	1



Shows distribution of ages, total number of entries (57 participants) and average: 27 years 10 months 6 days...

Histogram

- Constructor : `TH1 *h1 = new TH1("histName","histTitle",nofBins,minX,maxX)`
- Fill : `h1->Fill(value)`
- Fill with weight factor : `h1->Fill(value, weight)`
- Draw : `h1->Draw()`
- Scale : `h1->Sumw2() ; h1->Scale(factor)`
- Add : `h1->Add(h2)`
- Divide : `h1->Divide(h2)`
- Set x-axis range: `h1->SetAxisRange(2.,4.)`
- Set y-axis range: `h1->SetMinimum(2.); h1->SetMaximum(4.);`
- Merge 2 bins : `h1->Rebin(2)`
- Findbin : `h1->FindBin(3.0)`
- GetBinContent : `h1->GetBinContent(2); h1->GetBinContent(h1->FindBin(3.0))`

File

- Write objects to ROOT file :

```
TFile *fout = new TFile("output.root","recreate") ; obj->Write(); fout->Close(); delete fout;
```

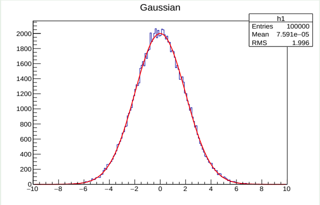
- Almost all ROOT classes are derived from TObject class, any types of objects that are defined inside the ROOT framework can be written (read) to (from) ROOT file. Generally written with ".root" extension.
- Any object created after the creation of TFile object will be written to the TFile (i.e. output.root in above example)
- If multiple ROOT files are opened, the last opened file will be used for writing the objects
- Read objects from ROOT file :

```
TFile *fin = new TFile("input.root") ; obj = fin->Get("obj_name");
```

- You can not close/delete the input file or input file pointer, if you want to use the "obj" pointer in the later part of your code

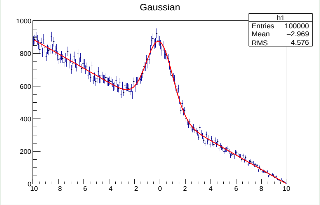
Fitting in ROOT : Gaussian

```
Double_t MyGaus(Double_t *x,Double_t *par) {
  Double_t arg = 0;
  if (par[2]!=0) arg = (x[0] - par[1])/par[2];
  Double_t fitval = par[0]*TMath::Exp(-0.5*arg*arg);
  return fitval;
}

int FitGaus()
{
  TH1F *h1 = new TH1F("h1","Gaussian",200,-10,10);
  TF1 *fn = new TF1("fn",MyGaus,-10,10,3);
  Double_t mean = 0.;
  Double_t sigma = 2.;
  Int_t N = 100000;
  for(int i = 0; i<N; i++) h1->Fill(gRandom->Gaus(mean,sigma));

  // fn->SetParameter(0,1);
  // fn->SetParameter(1,1);
  // fn->SetParameter(2,1);
  fn->SetParameters(1,1,1);

  TCanvas *c1 = new TCanvas("c1","c1");
  h1->Draw();
  //h1->Fit("gaus");
  h1->Fit("fn");
  cout<<"Mean : " <<fn->GetParameter(1)<<"+/-"<<fn->GetParError(1)<<endl;
  cout<<"Sigma : " <<fn->GetParameter(2)<<"+/-"<<fn->GetParError(2)<<endl;
  c1->Update();

  return 0;
}
```



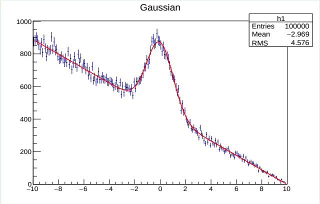
Fitting in ROOT : Signal + Bkg

```
#include <TF1.h>
#include <TH1F.h>
#include <TCanvas.h>
#include <iostream>
#include <TMath.h>
#include <TRandom.h>

using namespace std;

Double_t MyGaus(Double_t *x,Double_t *par) {
  Double_t arg = 0;
  if (par[2]!=0) arg = (x[0] - par[1])/par[2];
  Double_t fitval = par[0]*TMath::Exp(-0.5*arg*arg);
  return fitval;
}
Double_t MyBkg(Double_t *x,Double_t *par) {
  return par[0] + par[1]*x[0];
}
Double_t Total(Double_t *x,Double_t *par) {
  return par[0]*MyGaus(x,&par[2]) + par[1]*MyBkg(x,&par[5]);
}
```



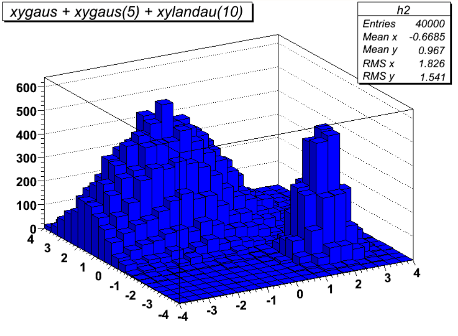
Fitting in ROOT : Signal + Bkg

```
int FitSignalBkg()
{
  TH1F *h1 = new TH1F("h1","Gaussian",200,-10,10);
  TF1 *fsig = new TF1("fsig",MyGaus,-10,10,3);
  TF1 *fbkg = new TF1("fbkg",MyBkg,-10,10,2);
  TF1 *ftot = new TF1("ftot",Total,-10,10,7);

  Double_t norm1 = 1.0,  norm2 = 1.;
  Double_t NGauss = 10., mean = 0., sigma = 1.;
  Double_t a = 10., b = -1.;

  ftot->SetParameters(norm1, norm2, NGauss, mean, sigma, a, b);

  Int_t N = 100000;
  for(int i = 0; i<N; i++) h1->Fill(ftot->GetRandom());

  ftot->SetParameters(1., 1., 1., 1., 1., 1., 1.);
  ftot->SetParLimits(4, 0.,3.); //sigma

  TCanvas *c1 = new TCanvas("c1","c1");
  h1->Draw("e");
  h1->Fit("ftot");

  Double_t *par, *parE;
  par = ftot->GetParameters();
  parE = ftot->GetParErrors();

  cout<<"Mean : " <<par[3]<<" +/- "<<parE[3]<<endl;
  cout<<"Sigma : " <<par[4]<<" +/- "<<parE[4]<<endl;

  return 0;
}
```

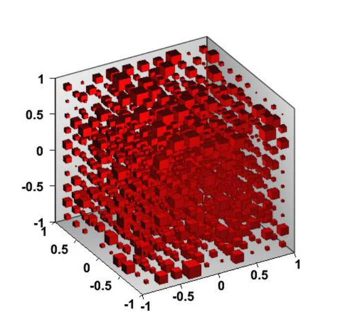


Graphics (CSC-2011)

We have seen 1D histograms, but there are also histograms in more dimensions.



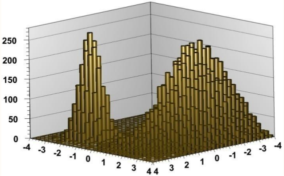
2D Histogram

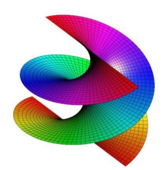


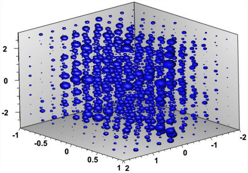
3D Histogram

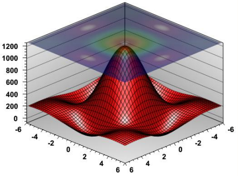
Graphics (CSC-2011)

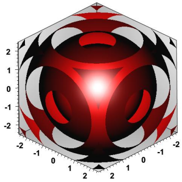
OpenGL can be used to render 2D & 3D histograms, functions, parametric equations, and to visualize 3D objects (geometry)











Checkpoint I

1. Start root session with splash screen
2. Add, subtract, multiply, divide
3. Redirect "all" output of ROOT session to temp.out file
4. Print out global environments to output.txt
5. gROOT, gSystem, gRandom, gPad, gStyle
6. List the methods of a class
7. Go to $ROOTSYS/tutorials, then apply .!pwd and pwd()
8. Try tab completion with edit("rootlogon.C")
9. Change the EDITOR environment and try again
10. Change back to earlier directory from $ROOTSYS/tutorials
11. Create class TPoint and print its' detail information
12. Set and print the variables in "for" loop inside ROOT session
13. Dump the object member values
14. Unnamed and named script : first.C vs rootlogon.C
15. Loading, unloading, running, compiling and compile+run
16. Compile in debug or optimized mode and +/++

Checkpoint I

1. Start root session with splash screen
2. Add, subtract, multiply, divide
3. Redirect “all” output of ROOT session to temp.out file
4. Print out global environments to output.txt
5. gROOT, gSystem, gRandom, gPad, gStyle
6. List the methods of a class
7. Go to $ROOTSYS/tutorials, then apply .!pwd and pwd()
8. Try tab completion with edit(”rootlogon.C”)
9. Change the EDITOR environment and try again
10. Change back to earlier directory from $ROOTSYS/tutorials
11. Create class TPoint and print its’ detail information
12. Set and print the variables in “for” loop inside ROOT session
13. Dump the object member values
14. Unnamed and named script : first.C vs rootlogon.C
15. Loading, unloading, running, compiling and compile+run
16. Compile in debug or optimized mode and +/++

Checkpoint I

1. Start root session with splash screen
2. Add, subtract, multiply, divide
3. Redirect "all" output of ROOT session to temp.out file
4. Print out global environments to output.txt
5. gROOT, gSystem, gRandom, gPad, gStyle
6. List the methods of a class
7. Go to $ROOTSYS/tutorials, then apply .!pwd and pwd()
8. Try tab completion with edit("rootlogon.C")
9. Change the EDITOR environment and try again
10. Change back to earlier directory from $ROOTSYS/tutorials
11. Create class TPoint and print its' detail information
12. Set and print the variables in "for" loop inside ROOT session
13. Dump the object member values
14. Unnamed and named script : first.C vs rootlogon.C
15. Loading, unloading, running, compiling and compile+run
16. Compile in debug or optimized mode and +/++

Checkpoint I

1. Start root session with splash screen
2. Add, subtract, multiply, divide
3. Redirect “all” output of ROOT session to temp.out file
4. Print out global environments to output.txt
5. gROOT, gSystem, gRandom, gPad, gStyle
6. List the methods of a class
7. Go to $ROOTSYS/tutorials, then apply .!pwd and pwd()
8. Try tab completion with edit(”rootlogon.C”)
9. Change the EDITOR environment and try again
10. Change back to earlier directory from $ROOTSYS/tutorials
11. Create class TPoint and print its’ detail information
12. Set and print the variables in “for” loop inside ROOT session
13. Dump the object member values
14. Unnamed and named script : first.C vs rootlogon.C
15. Loading, unloading, running, compiling and compile+run
16. Compile in debug or optimized mode and +/++

Checkpoint I

1. Start root session with splash screen
2. Add, subtract, multiply, divide
3. Redirect “all” output of ROOT session to temp.out file
4. Print out global environments to output.txt
5. gROOT, gSystem, gRandom, gPad, gStyle
6. List the methods of a class
7. Go to $ROOTSYS/tutorials, then apply .!pwd and pwd()
8. Try tab completion with edit("rootlogon.C")
9. Change the EDITOR environment and try again
10. Change back to earlier directory from $ROOTSYS/tutorials
11. Create class TPoint and print its' detail information
12. Set and print the variables in “for” loop inside ROOT session
13. Dump the object member values
14. Unnamed and named script : first.C vs rootlogon.C
15. Loading, unloading, running, compiling and compile+run
16. Compile in debug or optimized mode and +/++

Checkpoint I

1. Start root session with splash screen
2. Add, subtract, multiply, divide
3. Redirect “all” output of ROOT session to temp.out file
4. Print out global environments to output.txt
5. gROOT, gSystem, gRandom, gPad, gStyle
6. List the methods of a class
7. Go to $ROOTSYS/tutorials, then apply .!pwd and pwd()
8. Try tab completion with edit("rootlogon.C")
9. Change the EDITOR environment and try again
10. Change back to earlier directory from $ROOTSYS/tutorials
11. Create class TPoint and print its' detail information
12. Set and print the variables in “for” loop inside ROOT session
13. Dump the object member values
14. Unnamed and named script : first.C vs rootlogon.C
15. Loading, unloading, running, compiling and compile+run
16. Compile in debug or optimized mode and +/++

Checkpoint I

1. Start root session with splash screen
2. Add, subtract, multiply, divide
3. Redirect "all" output of ROOT session to temp.out file
4. Print out global environments to output.txt
5. gROOT, gSystem, gRandom, gPad, gStyle
6. List the methods of a class
7. Go to $ROOTSYS/tutorials, then apply .!pwd and pwd()
8. Try tab completion with edit("rootlogon.C")
9. Change the EDITOR environment and try again
10. Change back to earlier directory from $ROOTSYS/tutorials
11. Create class TPoint and print its' detail information
12. Set and print the variables in "for" loop inside ROOT session
13. Dump the object member values
14. Unnamed and named script : first.C vs rootlogon.C
15. Loading, unloading, running, compiling and compile+run
16. Compile in debug or optimized mode and +/++

Checkpoint I

1. Start root session with splash screen
2. Add, subtract, multiply, divide
3. Redirect "all" output of ROOT session to temp.out file
4. Print out global environments to output.txt
5. gROOT, gSystem, gRandom, gPad, gStyle
6. List the methods of a class
7. Go to $ROOTSYS/tutorials, then apply .!pwd and pwd()
8. Try tab completion with edit("rootlogon.C")
9. Change the EDITOR environment and try again
10. Change back to earlier directory from $ROOTSYS/tutorials
11. Create class TPoint and print its' detail information
12. Set and print the variables in "for" loop inside ROOT session
13. Dump the object member values
14. Unnamed and named script : first.C vs rootlogon.C
15. Loading, unloading, running, compiling and compile+run
16. Compile in debug or optimized mode and +/++

Checkpoint I

1. Start root session with splash screen
2. Add, subtract, multiply, divide
3. Redirect "all" output of ROOT session to temp.out file
4. Print out global environments to output.txt
5. gROOT, gSystem, gRandom, gPad, gStyle
6. List the methods of a class
7. Go to $ROOTSYS/tutorials, then apply .!pwd and pwd()
8. Try tab completion with edit("rootlogon.C")
9. Change the EDITOR environment and try again
10. Change back to earlier directory from $ROOTSYS/tutorials
11. Create class TPoint and print its' detail information
12. Set and print the variables in "for" loop inside ROOT session
13. Dump the object member values
14. Unnamed and named script : first.C vs rootlogon.C
15. Loading, unloading, running, compiling and compile+run
16. Compile in debug or optimized mode and +/++

Checkpoint I

1. Start root session with splash screen
2. Add, subtract, multiply, divide
3. Redirect "all" output of ROOT session to temp.out file
4. Print out global environments to output.txt
5. gROOT, gSystem, gRandom, gPad, gStyle
6. List the methods of a class
7. Go to $ROOTSYS/tutorials, then apply .!pwd and pwd()
8. Try tab completion with edit("rootlogon.C")
9. Change the EDITOR environment and try again
10. Change back to earlier directory from $ROOTSYS/tutorials
11. Create class TPoint and print its' detail information
12. Set and print the variables in "for" loop inside ROOT session
13. Dump the object member values
14. Unnamed and named script : first.C vs rootlogon.C
15. Loading, unloading, running, compiling and compile+run
16. Compile in debug or optimized mode and +/++

Checkpoint I

1. Start root session with splash screen
2. Add, subtract, multiply, divide
3. Redirect "all" output of ROOT session to temp.out file
4. Print out global environments to output.txt
5. gROOT, gSystem, gRandom, gPad, gStyle
6. List the methods of a class
7. Go to $ROOTSYS/tutorials, then apply .!pwd and pwd()
8. Try tab completion with edit("rootlogon.C")
9. Change the EDITOR environment and try again
10. Change back to earlier directory from $ROOTSYS/tutorials
11. Create class TPoint and print its' detail information
12. Set and print the variables in "for" loop inside ROOT session
13. Dump the object member values
14. Unnamed and named script : first.C vs rootlogon.C
15. Loading, unloading, running, compiling and compile+run
16. Compile in debug or optimized mode and +/++

Checkpoint I

1. Start root session with splash screen
2. Add, subtract, multiply, divide
3. Redirect “all” output of ROOT session to temp.out file
4. Print out global environments to output.txt
5. gROOT, gSystem, gRandom, gPad, gStyle
6. List the methods of a class
7. Go to $ROOTSYS/tutorials, then apply .!pwd and pwd()
8. Try tab completion with edit("rootlogon.C")
9. Change the EDITOR environment and try again
10. Change back to earlier directory from $ROOTSYS/tutorials
11. Create class TPoint and print its' detail information
12. Set and print the variables in “for” loop inside ROOT session
13. Dump the object member values
14. Unnamed and named script : first.C vs rootlogon.C
15. Loading, unloading, running, compiling and compile+run
16. Compile in debug or optimized mode and +/++

Checkpoint I

1. Start root session with splash screen
2. Add, subtract, multiply, divide
3. Redirect "all" output of ROOT session to temp.out file
4. Print out global environments to output.txt
5. gROOT, gSystem, gRandom, gPad, gStyle
6. List the methods of a class
7. Go to $ROOTSYS/tutorials, then apply .!pwd and pwd()
8. Try tab completion with edit("rootlogon.C")
9. Change the EDITOR environment and try again
10. Change back to earlier directory from $ROOTSYS/tutorials
11. Create class TPoint and print its' detail information
12. Set and print the variables in "for" loop inside ROOT session
13. Dump the object member values
14. Unnamed and named script : first.C vs rootlogon.C
15. Loading, unloading, running, compiling and compile+run
16. Compile in debug or optimized mode and +/++

Checkpoint I

1. Start root session with splash screen
2. Add, subtract, multiply, divide
3. Redirect “all” output of ROOT session to temp.out file
4. Print out global environments to output.txt
5. gROOT, gSystem, gRandom, gPad, gStyle
6. List the methods of a class
7. Go to $ROOTSYS/tutorials, then apply .!pwd and pwd()
8. Try tab completion with edit(”rootlogon.C”)
9. Change the EDITOR environment and try again
10. Change back to earlier directory from $ROOTSYS/tutorials
11. Create class TPoint and print its’ detail information
12. Set and print the variables in “for” loop inside ROOT session
13. Dump the object member values
14. Unnamed and named script : first.C vs rootlogon.C
15. Loading, unloading, running, compiling and compile+run
16. Compile in debug or optimized mode and +/++

Checkpoint I

1. Start root session with splash screen
2. Add, subtract, multiply, divide
3. Redirect “all” output of ROOT session to temp.out file
4. Print out global environments to output.txt
5. gROOT, gSystem, gRandom, gPad, gStyle
6. List the methods of a class
7. Go to $ROOTSYS/tutorials, then apply .!pwd and pwd()
8. Try tab completion with edit(”rootlogon.C”)
9. Change the EDITOR environment and try again
10. Change back to earlier directory from $ROOTSYS/tutorials
11. Create class TPoint and print its’ detail information
12. Set and print the variables in “for” loop inside ROOT session
13. Dump the object member values
14. Unnamed and named script : first.C vs rootlogon.C
15. Loading, unloading, running, compiling and compile+run
16. Compile in debug or optimized mode and +/++

Checkpoint I

1. Start root session with splash screen
2. Add, subtract, multiply, divide
3. Redirect "all" output of ROOT session to temp.out file
4. Print out global environments to output.txt
5. gROOT, gSystem, gRandom, gPad, gStyle
6. List the methods of a class
7. Go to $ROOTSYS/tutorials, then apply .!pwd and pwd()
8. Try tab completion with edit("rootlogon.C")
9. Change the EDITOR environment and try again
10. Change back to earlier directory from $ROOTSYS/tutorials
11. Create class TPoint and print its' detail information
12. Set and print the variables in "for" loop inside ROOT session
13. Dump the object member values
14. Unnamed and named script : first.C vs rootlogon.C
15. Loading, unloading, running, compiling and compile+run
16. Compile in debug or optimized mode and +/++

Checkpoint I

17. Write a class in a macro and run that prints "hello World"
18. How do you know the working directory inside macro ?
19. Where can you find earlier commands that have been applied during ROOT session ?
20. What is virtual function ? How to implement that in class ?
21. How to create abstract base class ?
22. Write code to create memory on stack and on heap
23. Write an example memory leak code

- How to compile macro containing ROOT classes using g++ ?
- Spot the memory leak in the code that you have written before.
- What is code profiling ? How does it help to improve your code ?

Checkpoint I

17 Write a class in a macro and run that prints "hello World"

18 How do you know the working directory inside macro ?

19 Where can you find earlier commands that have been applied during ROOT session ?

20 What is virtual function ? How to implement that in class ?

21 How to create abstract base class ?

22 Write code to create memory on stack and on heap

23 Write an example memory leak code

- How to compile macro containing ROOT classes using g++ ?
- Spot the memory leak in the code that you have written before.
- What is code profiling ? How does it help to improve your code ?

Checkpoint I

17. Write a class in a macro and run that prints "hello World"
18. How do you know the working directory inside macro ?
19. Where can you find earlier commands that have been applied during ROOT session ?
20. What is virtual function ? How to implement that in class ?
21. How to create abstract base class ?
22. Write code to create memory on stack and on heap
23. Write an example memory leak code

- How to compile macro containing ROOT classes using g++ ?
- Spot the memory leak in the code that you have written before.
- What is code profiling ? How does it help to improve your code ?

Checkpoint I

17 Write a class in a macro and run that prints “hello World”
18 How do you know the working directory inside macro ?
19 Where can you find earlier commands that have been applied during ROOT session ?
20 What is virtual function ? How to implement that in class ?
21 How to create abstract base class ?
22 Write code to create memory on stack and on heap
23 Write an example memory leak code

- How to compile macro containing ROOT classes using g++ ?
- Spot the memory leak in the code that you have written before.
- What is code profiling ? How does it help to improve your code ?

Checkpoint I

17. Write a class in a macro and run that prints "hello World"
18. How do you know the working directory inside macro ?
19. Where can you find earlier commands that have been applied during ROOT session ?
20. What is virtual function ? How to implement that in class ?
21. How to create abstract base class ?
22. Write code to create memory on stack and on heap
23. Write an example memory leak code

- How to compile macro containing ROOT classes using g++ ?
- Spot the memory leak in the code that you have written before.
- What is code profiling ? How does it help to improve your code ?

Checkpoint I

17. Write a class in a macro and run that prints "hello World"
18. How do you know the working directory inside macro ?
19. Where can you find earlier commands that have been applied during ROOT session ?
20. What is virtual function ? How to implement that in class ?
21. How to create abstract base class ?
22. Write code to create memory on stack and on heap
23. Write an example memory leak code

- How to compile macro containing ROOT classes using g++ ?
- Spot the memory leak in the code that you have written before.
- What is code profiling ? How does it help to improve your code ?

Checkpoint I

17. Write a class in a macro and run that prints “hello World”
18. How do you know the working directory inside macro ?
19. Where can you find earlier commands that have been applied during ROOT session ?
20. What is virtual function ? How to implement that in class ?
21. How to create abstract base class ?
22. Write code to create memory on stack and on heap
23. Write an example memory leak code

- How to compile macro containing ROOT classes using g++ ?
- Spot the memory leak in the code that you have written before.
- What is code profiling ? How does it help to improve your code ?

Checkpoint I

17. Write a class in a macro and run that prints "hello World"
18. How do you know the working directory inside macro ?
19. Where can you find earlier commands that have been applied during ROOT session ?
20. What is virtual function ? How to implement that in class ?
21. How to create abstract base class ?
22. Write code to create memory on stack and on heap
23. Write an example memory leak code

- How to compile macro containing ROOT classes using g++ ?
- Spot the memory leak in the code that you have written before.
- What is code profiling ? How does it help to improve your code ?

Checkpoint I

17. Write a class in a macro and run that prints “hello World”
18. How do you know the working directory inside macro ?
19. Where can you find earlier commands that have been applied during ROOT session ?
20. What is virtual function ? How to implement that in class ?
21. How to create abstract base class ?
22. Write code to create memory on stack and on heap
23. Write an example memory leak code

- How to compile macro containing ROOT classes using g++ ?
- Spot the memory leak in the code that you have written before.
- What is code profiling ? How does it help to improve your code ?

Checkpoint I

17. Write a class in a macro and run that prints "hello World"
18. How do you know the working directory inside macro ?
19. Where can you find earlier commands that have been applied during ROOT session ?
20. What is virtual function ? How to implement that in class ?
21. How to create abstract base class ?
22. Write code to create memory on stack and on heap
23. Write an example memory leak code

- How to compile macro containing ROOT classes using g++ ?
- Spot the memory leak in the code that you have written before.
- What is code profiling ? How does it help to improve your code ?

Tree

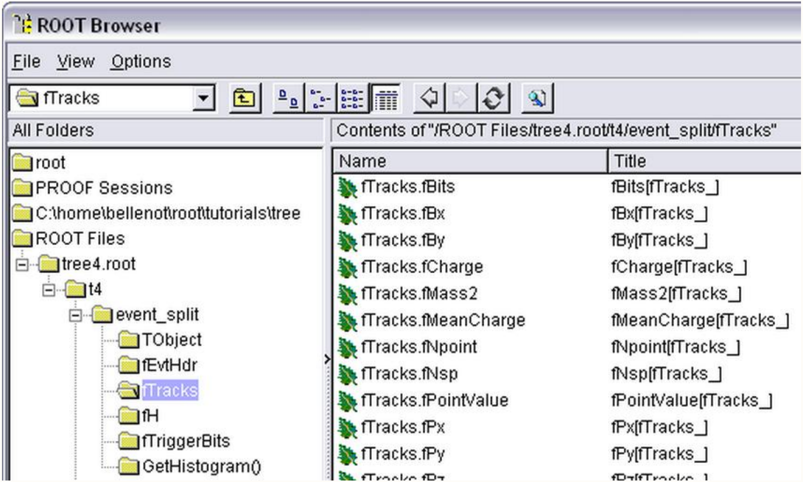
- Arrange different types of objects and data types in single place.
- Formatted in such way such that accessing the entries is fast.
- While written to disk uses less disk resource

Tree (CSC-2011)

Tree structure

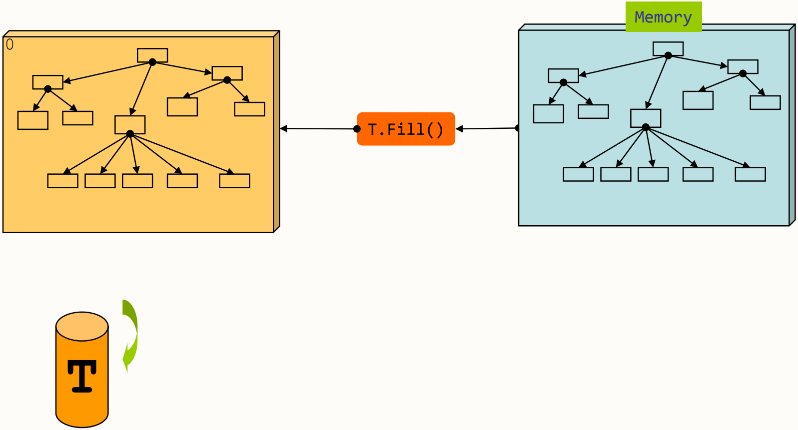
- Branches: directories
- Leaves: data containers
- Can read a subset of all branches – speeds up considerably the data analysis processes
- Branches of the same **TTree** can be written to separate files

Tree structure



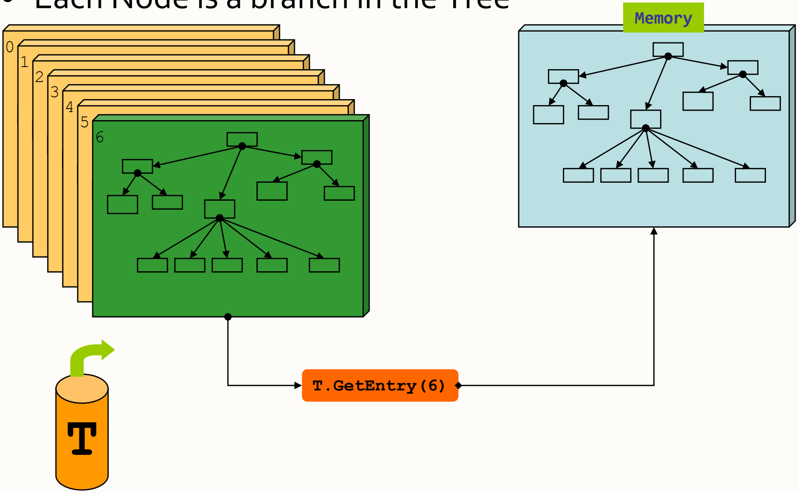
Memory ↔ Tree

- Each Node is a branch in the Tree



Memory ↔ Tree

- Each Node is a branch in the Tree



Five Steps to Build a Tree

Steps:

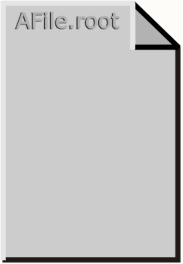
1. Create a TFile
2. Create a TTree
3. Add TBranch to the TTree
4. Fill the tree
5. Write the file

Example macro

```
void WriteTree()
{
   Event *myEvent = new Event();
   TFile f("AFile.root", "RECREATE");
   TTree *t = new TTree("myTree","A Tree");
   t->Branch("EventBranch", &myEvent);
   for (int e=0;e<100000;++e) {
      myEvent->Generate();  // hypothetical
      t->Fill();
   }
   t->Write();
}
```

Step 1: Create a TFile Object

Trees can be huge → need file for swapping filled entries

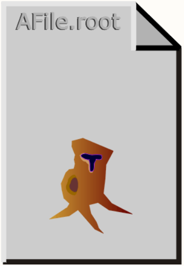



```
TFile *hfile = TFile::Open("AFile.root",
                           "RECREATE");
```

Step 2: Create a TTree Object

The TTree constructor:

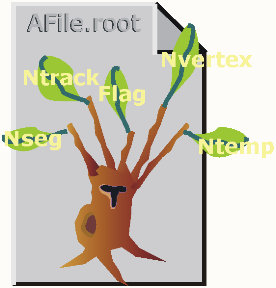
- Tree name (e.g. "myTree")
- Tree title




```
TTree *tree = new TTree("myTree","A Tree");
```

Step 3: Adding a Branch

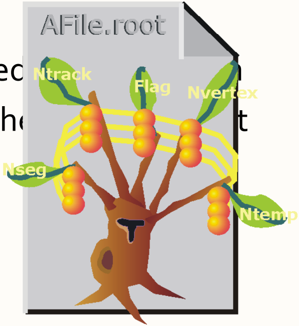
- Branch name
- Address of pointer
  to the object




```
Event *myEvent = new Event();
myTree->Branch("eBranch", &myEvent);
```

Step 4: Fill the Tree

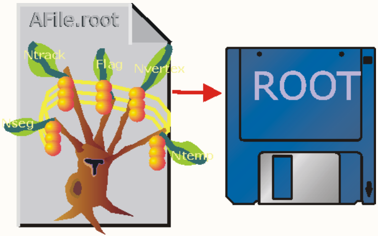
- Create a for loop
- Assign values to the object contained
- TTree::Fill() creates a new entry in the
  of values of branches' objects




```
for (int e=0;e<100000;++e) {
   myEvent->Generate(e); // fill event
   myTree->Fill();       // fill the tree
}
```

Step 5: Write Tree To File

```
myTree->Write();
```



Reading a TTree

- Looking at a tree
- How to read a tree
- Friends and chains

Example macro

```
void ReadTree() {
  TFile f("AFile.root");
  TTree *T = (TTree*)f->Get("T");
  Event *myE = 0; TBranch* brE = 0;
  T->SetBranchAddress("EvBranch", &myE, brE);
  T->SetCacheSize(10000000);
  T->AddBranchToCache("EvBranch");
  Long64_t nbent = T->GetEntries();
  for (Long64_t e = 0;e < nbent; ++e) {
     brE->GetEntry(e);
     myE->Analyze();
  }
}
```

Data pointers (e.g. myE) MUST be set to 0

How to Read a TTree

Example:

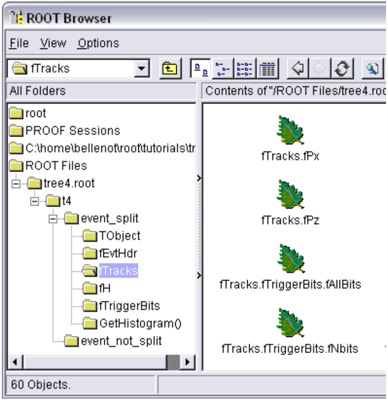
1. Open the Tfile

```
TFile f("AFile.root")
```

2. Get the TTree

```
TTree *myTree = 0;
f.GetObject("myTree",myTree)
```

or



How to Read a TTree

3. Create a variable pointing to the data

```
root [] Event *myEvent = 0;
```

4. Associate a branch with the variable:

```
root [] myTree->SetBranchAddress("eBranch", &myEvent);
```

5. Read one entry in the TTree

```
root [] myTree->GetEntry(0)
root [] myEvent->GetTracks()->First()->Dump()
==> Dumping object at: 0x0763aad0, name=Track, class=Track
fPx             0.651241    X component of the momentum
fPy             1.02466     Y component of the momentum
fPz             1.2141      Z component of the momentum
[...]
```

Branch Access Selection

- Use TTree::SetBranchStatus() or TBranch::GetEntry() to select branches to be read
- Speed up considerably the reading phase

```
TClonesArray* myMuons = 0;
// disable all branches
myTree->SetBranchStatus("*", 0);
// re-enable the "muon" branches
myTree->SetBranchStatus("muon*", 1);
myTree->SetBranchAddress("muon", &myMuons);
// now read (access) only the "muon" branches
myTree->GetEntry(0);
```

Looking at the Tree

TTree::Print() shows the data layout

```
root [] TFile f("AFile.root")
root [] myTree->Print();
******************************************************************************
*Tree    :myTree    : A ROOT tree                                            *
*Entries :       10 : Total =          867935 bytes  File  Size =     390138 *
*        :          : Tree compression factor =   2.72                       *
******************************************************************************
*Branch  :eBranch                                                        *
*Entries :       10 : BranchElement (see below)                              *
*............................................................................*
*Br    0 :fUniqueID :                                                        *
*Entries :       10 : Total  Size=        698 bytes  One basket in memory    *
*Baskets :        0 : Basket Size=      64000 bytes  Compression=   1.00     *
*............................................................................*
…
…
```

TTree Selection Syntax

Print the first 8 variables of the tree:

```
MyTree->Scan();
```

Prints all the variables of the tree:

```
MyTree->Scan("*");
```

Prints the values of var1, var2 and var3.

```
MyTree->Scan("var1:var2:var3");
```

A selection can be applied in the second argument:

```
MyTree->Scan("var1:var2:var3", "var1>0");
```

Prints the values of var1, var2 and var3 for the entries where var1 is greater than 0

Use the same syntax for TTree::Draw()

Looking at the Tree

TTree::Show(entry_number) shows values for one entry

```
root [] myTree->Show(0);
======> EVENT:0
eBranch          = NULL
fUniqueID        = 0
fBits            = 50331648
[...]
fNtrack          = 594
fNseg            = 5964
[...]
fEvtHdr.fRun     = 200
[...]
fTracks.fPx      = 2.066806, 0.903484, 0.695610,-0.637773,…
fTracks.fPy      = 1.459911, -0.409338, 0.391340, 1.244357,…
```

TChain: the Forest

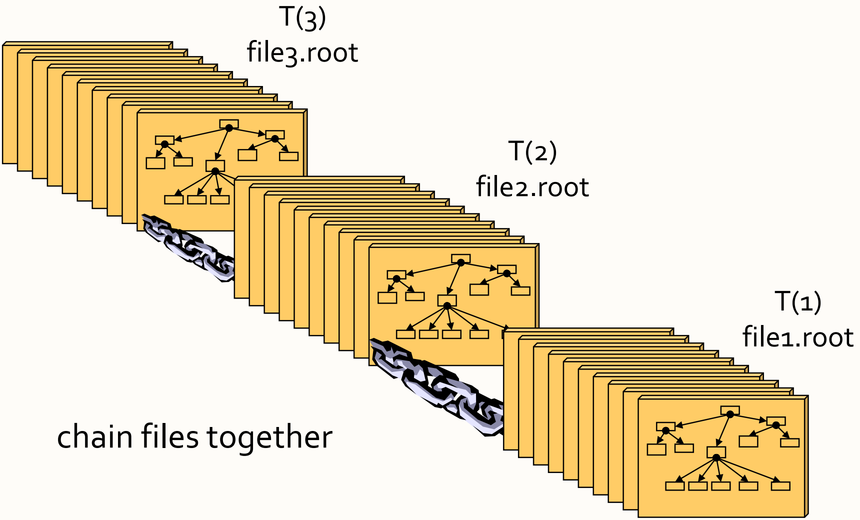
- Collection of TTrees: list of ROOT files containing the same tree
- Same semantics as TTree

As an example, assume we have three files called file1.root, file2.root, file3.root. Each contains tree called "T". Create a chain:

```
TChain chain("T"); // argument: tree name
chain.Add("file1.root");
chain.Add("file2.root");
chain.Add("file3.root");
```

Now we can use the TChain like a TTree!

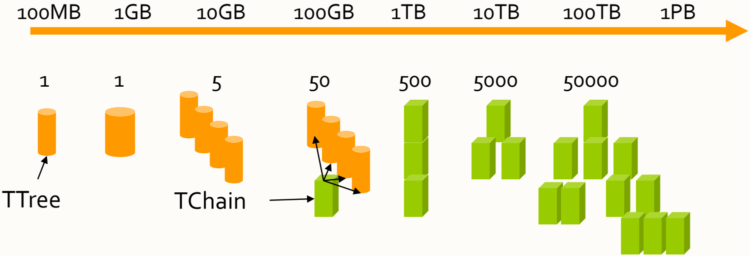
TChain



chain files together

Data Volume & Organisation

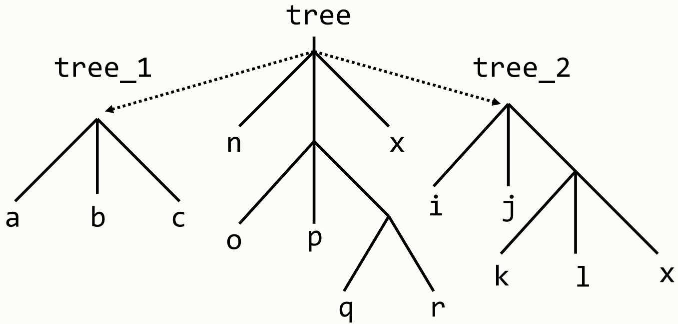
- A TFile typically contains 1 TTree
- A TChain is a collection of TTrees or/and TChains



Tree Friends

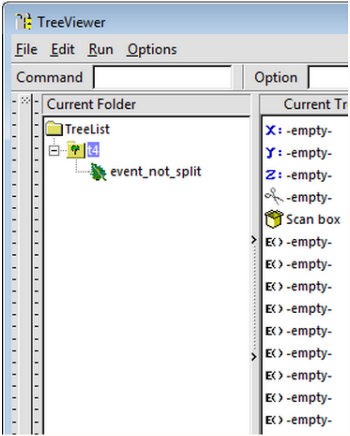
- Trees are designed to be read only
- Often, people want to add branches to existing trees and write their data into it
- Using tree friends is the solution:
  - Create a new file holding the new tree
  - Create a new Tree holding the branches for the user data
  - Fill the tree/branches with user data
  - Add this new file/tree as friend of the original tree

Tree Friends

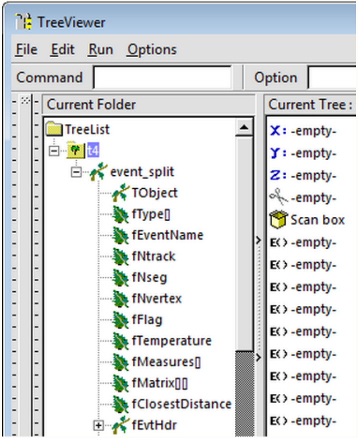



```
TFile f1("tree.root");
tree.AddFriend("tree_1", "tree1.root")
tree.AddFriend("tree_2", "tree2.root");
tree.Draw("x:a", "k<c");
tree.Draw("x:tree_2.x");
```

Splitting



Split level = 0



Split level = 99

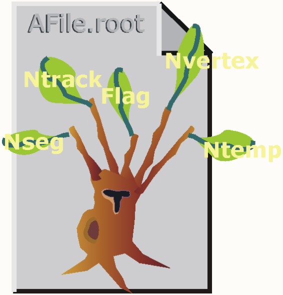
Splitting

- Creates one branch per member – recursively
- Allows to browse objects that are stored in trees, even without their library
- Fine grained branches allow fine-grained I/O - read only members that are needed
- Supports STL containers too, even vector<T*>!

Splitting

Setting the split level (default = 99)

Split level = 0



Split level = 99

```
tree->Branch("EvBr", &event, 64000, 0 );
```

Performance Considerations

A split branch is:

- Faster to read – if you only want a subset of data members
- Slower to write due to the large number of branches

Summary: Trees

- TTree is one of the most powerful collections available for HEP
- Extremely efficient for huge number of data sets with identical layout
- Very easy to look at TTree - use TBrowser!
- Write once, read many (WORM) ideal for experiments' data; use friends to extend
- Branches allow granular access; use splitting to create branch for each member, even through collections

Tree

```
class Det {  // each detector gives an energy and time signal
public:
  Double_t e; //energy
  Double_t t; //time

//  ClassDef(Det,1)
};
```

```
class Event : public TObject {
public:

  Det a; // say there are two detectors (a and b) in the experiment
  Det b;
  ClassDef(Event,1)
};
```

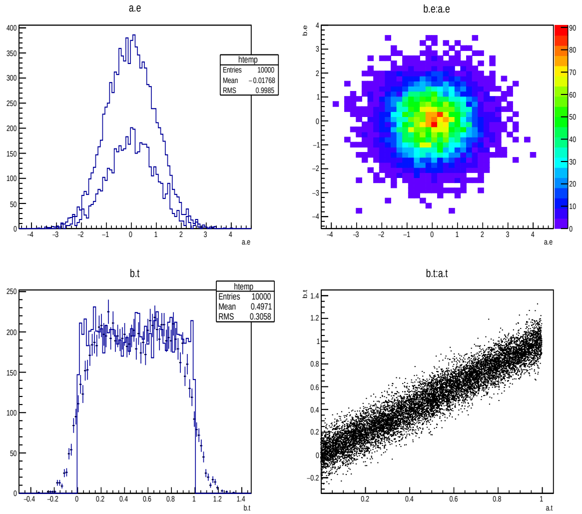
Tree

```
TTree *tree = new TTree("tree","treelibrated tree");
Event *e = new Event;

// create a branch with energy
tree->Branch("event",&e);

// fill some events with random numbers
Int_t nevent=10000;
for (Int_t iev=0;iev<nevent;iev++) {
  if (iev%1000==0) cout<<"Processing event "<<iev<<"..."<<endl;

  Float_t ea,eb;
  gRandom->Rannor(ea,eb); // the two energies follow a gaus distribution
  e->a.e=ea;
  e->b.e=eb;
  e->a.t=gRandom->Rndm();  // random
  e->b.t=e->a.t + gRandom->Gaus(0.,.1);  // identical to a.t but a gaussian
                                         // 'resolution' was added with sigma .1

  tree->Fill();  // fill the tree with the current event
}
```

Tree

```
// now draw some tree variables
TCanvas *c1 = new TCanvas();
c1->Divide(2,2);
c1->cd(1);
tree->Draw("a.e");  //energy of det a
tree->Draw("a.e","3*(-.2<b.e && b.e<.2)","same");  // same but with condition
c1->cd(2);
tree->Draw("b.e:a.e","","colz");        // one energy against the other
c1->cd(3);
tree->Draw("b.t","","e");    // time of b with errorbars
tree->Draw("a.t","","same"); // overlay time of detector a
c1->cd(4);
tree->Draw("b.t:a.t");       // plot time b again time a
```

Tree



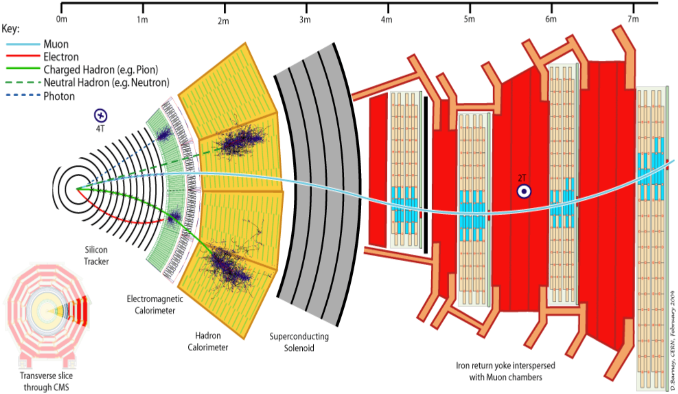
a.e
htemp
Entries 10000
Mean −0.01768
RMS 0.9985
b.e:a.e
b.e
a.e
b.t
htemp
Entries 10000
Mean 0.4971
RMS 0.3058
b.t:a.t
b.t
a.t

HEP events (Kinematics)

- It starts with a collision.
- Different sets of particles are produced at various stages.
- High energetic particles are favoured at the earlier stage of collisions.
- It is then followed by the generation of low mass particles.
- The unstable particles decay into the stable particles in single or multiple steps.

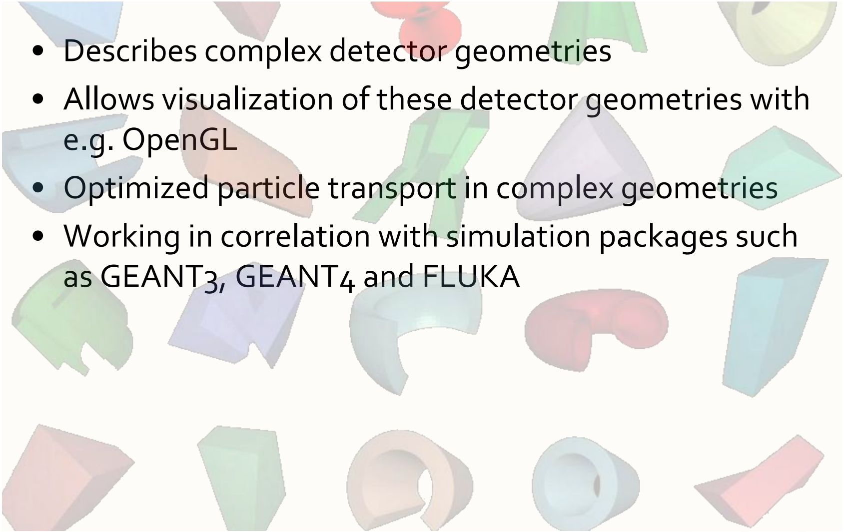
HEP events (Detector)

We measure the stable particles that interacts with the detector. The particle transport code like GEANT, FLUKA takes care of the interaction processes.

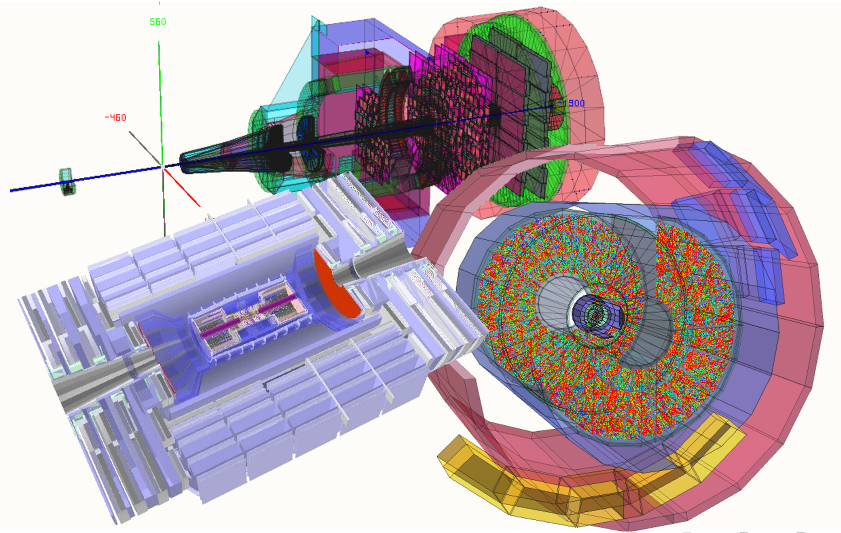


Graphics (CSC-2011)

- Describes complex detector geometries
- Allows visualization of these detector geometries with e.g. OpenGL
- Optimized particle transport in complex geometries
- Working in correlation with simulation packages such as GEANT3, GEANT4 and FLUKA



Graphics (CSC-2011)

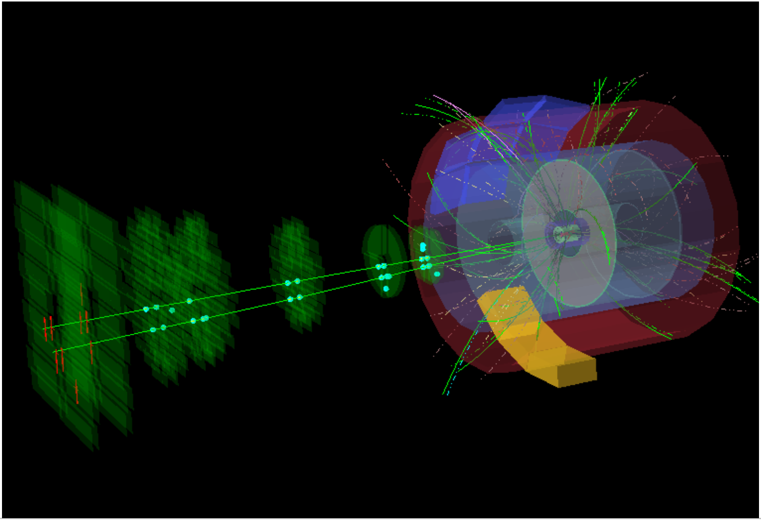


HEP events (Reconstruction)

- Find out the detector channels that have been fired.
- Apply the detector calibration and find out the corrected charge deposition.
- Combine the information from Calorimeter and Tracking detectors.
- Reconstruct the track of particles which provides the information of (p_x, p_y, p_z, E)
- Combine the tracks of secondary particles to find out the primary particles.

Event display of pp collision

event → track → cluster



Track header containing cluster

Compare with the classes in the TTree section

```
class Track : public TObject {

private:
   Float_t      fPx;           //X component of the momentum
   Float_t      fPy;           //Y component of the momentum
   Float_t      fPz;           //Z component of the momentum
   Float_t      fRandom;       //A random track quantity
   Float16_t    fMass2;        //[0,0,8] The mass square of this particle
   Float16_t    fBx;           //[0,0,10] X intercept at the vertex
   Float16_t    fBy;           //[0,0,10] Y intercept at the vertex
   Float_t      fMeanCharge;   //Mean charge deposition of all hits of this track
   Float16_t    fXfirst;       //X coordinate of the first point
   Float16_t    fXlast;        //X coordinate of the last point
   Float16_t    fYfirst;       //Y coordinate of the first point
   Float16_t    fYlast;        //Y coordinate of the last point
   Float16_t    fZfirst;       //Z coordinate of the first point
   Float16_t    fZlast;        //Z coordinate of the last point
   Double32_t   fCharge;       //[-1,1,2] Charge of this track
   Double32_t   fVertex[3];    //[-30,30,16] Track vertex position
   Int_t        fNpoint;       //Number of points for this track
   Short_t      fValid;        //Validity criterion
   Int_t        fNsp;          //Number of points for this track with a special value
   Double32_t*  fPointValue;   //[fNsp][0,3] a special quantity for some point.
   TBits        fTriggerBits;  //Bits triggered by this track.

public:
   Track() : fTriggerBits(64) { fNsp = 0; fPointValue = 0; }
   Track(const Track& orig);
   Track(Float_t random);
   virtual ~Track() {Clear();}
```

Event header

```
class Event : public TObject {

private:
   char            fType[20];           //event type
   char           *fEventName;          //run+event number in character format
   Int_t           fNtrack;             //Number of tracks
   Int_t           fNseg;               //Number of track segments
   Int_t           fNvertex;
   UInt_t          fFlag;
   Double32_t      fTemperature;
   Int_t           fMeasures[10];
   Double32_t      fMatrix[4][4];
   Double32_t     *fClosestDistance;    //[fNvertex][0,0,6]
   EventHeader     fEvtHdr;
   TClonesArray   *fTracks;             //->array with all tracks
   TRefArray      *fHighPt;             //array of High Pt tracks only
   TRefArray      *fMuons;              //array of Muon tracks only
   TRef            fLastTrack;          //reference pointer to last track
   TRef            fWebHistogram;       //EXEC:GetWebHistogram reference to an
   TH1F           *fH;                  //->
   TBits           fTriggerBits;        //Bits triggered by this event.
   Bool_t          fIsValid;            //

   static TClonesArray *fgTracks;
   static TH1F         *fgHist;

public:
   Event();
   virtual ~Event();
   void           Build(Int_t ev, Int_t arg5=600, Float_t ptmin=1);
   void           Clear(Option_t *option ="");
   Bool_t         IsValid() const { return fIsValid; }
   static void    Reset(Option_t *option ="");
```

Writing HEP event

Splitting or not splitting

```
//create a Tree file tree.root
TFile f("tree.root","RECREATE");

// Create a ROOT Tree
TTree tree("tree","A Tree with Events");

// Create a pointer to an Event object
Event *event = new Event();

// Create two branches, split one.
tree.Branch("event_split", &event,16000,99);
tree.Branch("event_not_split", &event,16000,0);

// a local variable for the event type
char etype[20];
```

Writing HEP event

```
for (Int_t ev = 0; ev <100; ev++) {
  Float_t sigmat, sigmas;
  gRandom->Rannor(sigmat,sigmas);
  Int_t ntrack   = Int_t(600 + 600 *sigmat/120.);
  Float_t random = gRandom->Rndm(1);
  sprintf(etype,"type%d",ev%5);
  event->SetType(etype);
  event->SetHeader(ev, 200, 960312, random);
  event->SetNseg(Int_t(10*ntrack+20*sigmas));
  event->SetNvertex(Int_t(1+20*gRandom->Rndm()));
  event->SetFlag(UInt_t(random+0.5));
  event->SetTemperature(random+20.);
  for(UChar_t m = 0; m < 10; m++) {
    event->SetMeasure(m, Int_t(gRandom->Gaus(m,m+1)));
  }

  for(UChar_t i0 = 0; i0 < 4; i0++) {
    for(UChar_t i1 = 0; i1 < 4; i1++) {
      event->SetMatrix(i0,i1,gRandom->Gaus(i0*i1,1));
    }
  }

  for (Int_t t = 0; t < ntrack; t++) event->AddTrack(random);

  tree.Fill();

  event->Clear();
}

f.Write();
```

Reading HEP event

```
TFile *f = new TFile("tree.root");
TTree *tree = (TTree*)f->Get("tree");

Event *event = new Event();

// get two branches and set the branch address
TBranch *bntrack = tree->GetBranch("fNtrack");
TBranch *branch  = tree->GetBranch("event_split");
branch->SetAddress(&event);

Long64_t nevent = tree->GetEntries();
Int_t nselected = 0;
Int_t nb = 0;
for (Long64_t i=0;i<nevent;i++) {
  //read branch "fNtrack"only
  bntrack->GetEntry(i);

  //reject events with more than 587 tracks
  if (event->GetNtrack() > 587)continue;

  //read complete accepted event in memory
  nb += tree->GetEntry(i);
  nselected++;

  //print the first accepted event
  if (nselected == 1) tree->Show();

  //clear tracks array
  event->Clear();
}
```

Checkpoint II

1. In case multiple files are opened, how the object can be written in the first file instead of last ?
2. In which case the input ROOT file can be closed while you are still using the object stored into that file ?
3. Write/read the event tree to/from ROOT file
4. Now scan tree.root for fNtrack and fNvertex in ROOT session
5. Next scan tree.root for fNTracks->fPx and fNvertex in ROOT session
6. Draw fNTracks->fPx from tree.root in ROOT session using tree->Draw("")
7. Draw fNTracks->fPx vs fNTracks->fPz from tree.root in ROOT session
8. Draw fNTracks->fPx vs fNTracks->fPz for (fNvertex>5) in ROOT session

- Copy the tree.root of above example into tree1.root, tree2.root and tree3.root and read all three files using TChain in a macro.

ALICE software installation

- Read the detail installatation process at,
  https://alice-doc.github.io/alice-analysis-tutorial/building/
- Start with simpler installation method "Install ALICE software with alibuild".
- However, if you own a laptop you may try the quick start procedure.
- The terminal logs of quick start procedure are uploaded in the indico page.
- The complete build procedure,
  - i took ~12 hours, in a typical Indian home network.
  - ii The complete build procedure will ask you few times to apply your CERN credentials.
  - iii The complete build procedure may require ~24 GB of diskspace.
- Note that this quick start is a standalone procedure. The software discussions with the collaboration colleagues should be based on the information ath the link as mentioned above.

ALICE software installation

Install the python and alibuild package as superuser 'root'

```
# yum install python-pip
# pip install –upgrade pip
# pip install alibuild
```

I presume you install ALICE software in $HOME/alice

```
$ mkdir $HOME/alice
$ cd $HOME/alice
$ export ALIBUILD_WORK_DIR="$HOME/alice/sw"
$ aliBuild init AliRoot,AliPhysics (this will download the git repository)
$ aliDoctor AliPhysics (download the packages required for installations as
mentioned in the output)
$ aliBuild build AliPhysics
$ export ALICE_WORK_DIR="$HOME/alice/sw"
$ alienv q
$ alienv enter AliPhysics/latest
```

ALICE software installation

If you want to keep another version of AliPhysics and AliRoot, without changing the previous installation,

```
$ mkdir $HOME/alice/v5-09-02-01
$ cd $HOME/alice/v5-09-02-01
$ aliBuild init AliRoot,AliPhysics -w ../sw
$ cd AliPhysics ; git checkout v5-09-02-01 ; cd ../
$ cd AliRoot ; git checkout v5-09-02 ; cd ..
$ aliBuild build AliPhysics -w ../sw -z
$ alienv q
$ alienv enter VO_ALICE@AliPhysics::latest-v5-09-02-01-release
```

- http://alimonitor.cern.ch/packages/

THANK YOU