ROOT

Documentation about ROOT

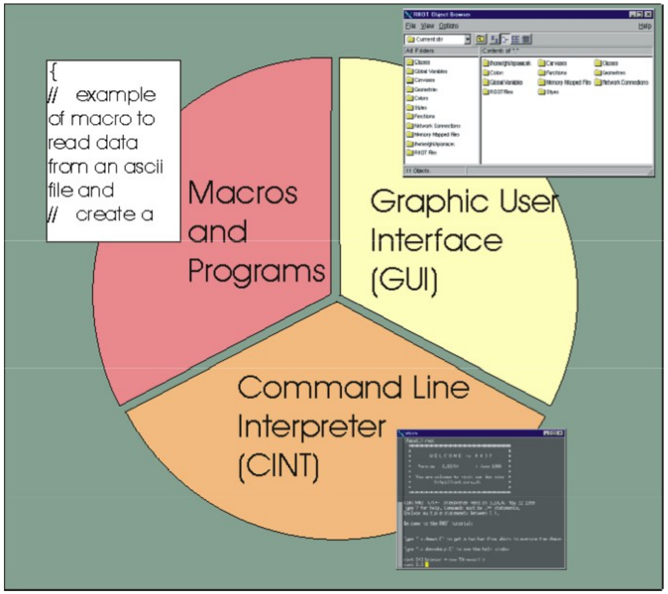
- www.root.cern.ch
  - Tutorials, user's guide, class reference guide, FAQs, how to's, ...

Contents

- Root basics
- Opening/closing files
  - Persistency in ROOT
- Histograms
- Storing data
- Library of classes and tools for data analysis and processing
- Exercises

What is ROOT?

- Framework for data analysis based on C++ (Object Oriented)
  - Store/read data
  - Process data
  - Analyse the data (histograming, statistical treatment, …)
- Developed and supported by the High Energy Physics community
  - www.root.cern.ch
- Widely used
- Tree possible ways to use it
  - Graphical User Interface
  - CINT: C++ interpreter
  - Macros, programs, libraries
- Can be used as a collection of libraries



The CINT interpreter

- The most basic way to use ROOT
  - Can execute instruction by instruction
  - Can be used interactively

```
pconde — root.exe — 80×24
[Amelias-MacBook-Air]: ~ > root
  *******************************************
  *                                         *
  *        W E L C O M E  to  R O O T       *
  *                                         *
  *   Version   5.34/01      13 July 2012   *
  *                                         *
  *  You are welcome to visit our Web site  *
  *          http://root.cern.ch            *
  *                                         *
  *******************************************

ROOT 5.34/01 (tags/v5-34-01@45048, Jul 13 2012, 15:31:31 on macosx64)

CINT/ROOT C/C++ Interpreter version 5.18.00, July 2, 2010
Type ? for help. Commands must be C++ statements.
Enclose multiple statements between { }.
root [0] int test = 6
root [1] test
(int)6
root [2] cout << "test = " << test
test = 6(class ostream)140735252709232
root [3] cout << "test = " << test;
test = 6root [4]
```

Many, many, many different objects

- TH1, TH1F, TH2, TH2F, … : histograms
- TFiles: object to handle input/output files
- TCanvas: object to handle screens to plot histograms or other graphical objects
- Ttrees, Tntuples: objects to structure data
- TGraph: object to store sets of measurement points
- TVector,TMatrix, TRotation, …
- TFormula: object to create your own mathematical functions
- TFit: object to do fits
- …
- No need to know all of them: will use these objects as a library that provides functionality for our simulations

✯ Example

Function Members (Methods)

public:

	TFile ()
	TFile (const char* fname, Option_t* option = "", const char* ftitle = "", Int_t compress = 1)
virtual	**~TFile** ()
void	TObject::**AbstractMethod** (const char* method) const
virtual void	TDirectoryFile::**Add** (TObject* obj, Bool_t replace = kFALSE)
static void	TDirectory::**AddDirectory** (Bool_t add = kTRUE)
static Bool_t	TDirectory::**AddDirectoryStatus** ()
virtual void	TDirectoryFile::**Append** (TObject* obj, Bool_t replace = kFALSE)
virtual Int_t	TDirectoryFile::**AppendKey** (TKey* key)
virtual void	TObject::**AppendPad** (Option_t* option = "")
static TFileOpenHandle*	**AsyncOpen** (const char* name, Option_t* option = "", const char* ftitle = "", Int_t compress = netopt = 0)
virtual void	TDirectoryFile::**Browse** (TBrowser* b)
virtual void	TDirectoryFile::**Build** (TFile* motherFile = 0, TDirectory* motherDir = 0)
virtual Bool_t	TDirectoryFile::**cd** (const char* path = 0)
static Bool_t	TDirectory::**Cd** (const char* path)

class TFile -

library: libRIO
#include "TFile.h"

Display options:
☑ Show inherited
☑ Show non-public

[↑ Top] | [? Help]

ROOT Memory management

- ROOT objects (Histograms, Canvas, etc) are managed in memory (and disk) by root using "names"
- In the same directory you cannot have two object with the same name (ROOT will complain about memory leaks)
  - ROOT does not like the following

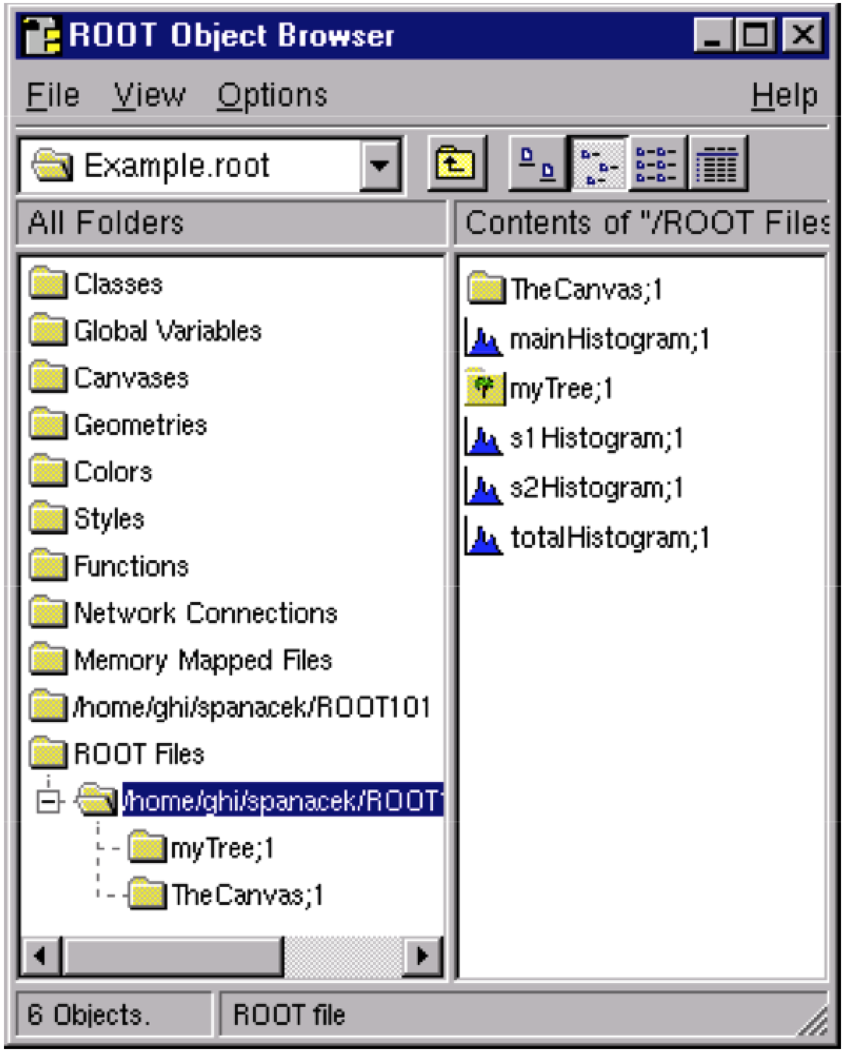
Same name!

```
TH1F * histos[10];
for(int i = 0; i < 10 ; i++) histos[i]= new TH1F("hist","hist",1,2,3);
```

- Objects member functions can be accessed with "." (for instance and reference) or "->" (for pointers) root ""understand" both
  - Interactive ROOT fixes for you wrong usage of pointer vs reference, but when you compile you MUST use the correct syntax.

File structure

- A directory structure like UNIX
- Items in a directory can be:
  - Subdirectory
  - Objects (canvas, histograms, etc.)



Opening/closing files

- Opening files using the command line (CINT interpreter)

```
  *******************************************

 ROOT 5.34/01 (tags/v5-34-01@45048, Jul 13 2012, 15:31:31 on macosx64)

 CINT/ROOT C/C++ Interpreter version 5.18.00, July 2, 2010
 Type ? for help. Commands must be C++ statements.
 Enclose multiple statements between { }.
 root [0] TFile myFile("TestFile.root","READ")
 root [1] myFile.ls()
 TFile**         TestFile.root
  TFile*         TestFile.root
   KEY: TH1F     hGaus;1
 root [2]
```

- Accessing the histogram stored on the file:

```
root [2] TH1F *h1 = (TH1F *) myFile.Get("hGaus")
root [3] h1->Draw()
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
root [4]
```

Opening/closing files

- Opening files using the command line (CINT interpreter)

```
  *******************************************

ROOT 5.34/01 (tags/v5-34-01@45048, Jul 13 2012, 15:31:31 on macosx64)

CINT/ROOT C/C++ Interpreter version 5.18.00, July 2, 2010
Type ? for help. Commands must be C++ statements.
Enclose multiple statements between { }.
root [0] TFile myFile("TestFile.root","READ")
root [1] myFile.ls()
TFile**         TestFile.root
 TFile*         TestFile.root
  KEY: TH1F     hGaus;1
root [2]
```

- Accessing the histogram stored on th

```
root [2] TH1F *h1 = (TH1F *) myFile.Get("hGaus")
root [3] h1->Draw()
Info in <TCanvas::MakeDefCanvas>:  created default
root [4]
```

- Create a pointer of type TH1F (histogram 1D, float)
- Retrieve the object "hGaus" from the file
- Dynamic cast the output to a TH1F (it is a generic type TObject)
- Initialize pointer to the retrieved object

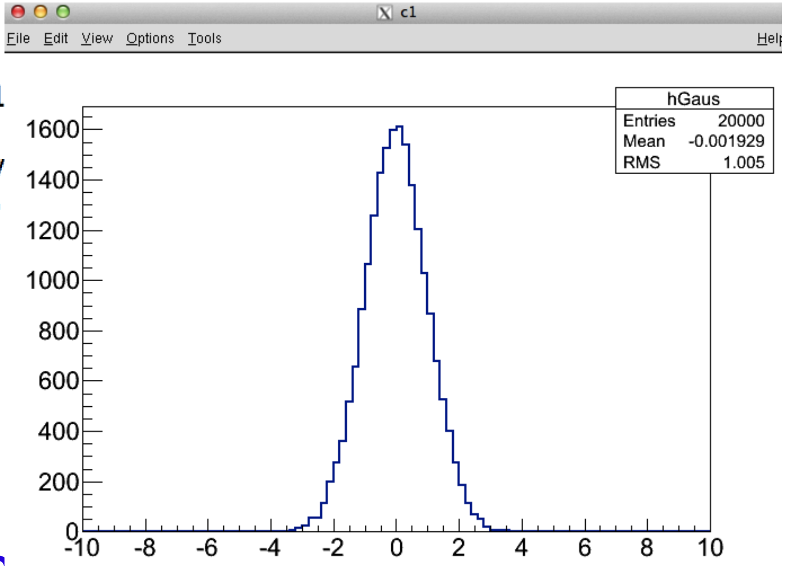
Opening/closing files

- Opening files using the command line (CINT interpreter)

```
  ********************************************

 ROOT 5.34/01 (tags/v5-34-01@45048, Jul 13 2012, 1

 CINT/ROOT C/C++ Interpreter version 5.18.00, July
 Type ? for help. Commands must be C++ statements.
 Enclose multiple statements between { }.
 root [0] TFile myFile("TestFile.root","READ")
 root [1] myFile.ls()
 TFile**         TestFile.root
  TFile*         TestFile.root
   KEY: TH1F     hGaus;1
 root [2]
```



- Accessing the histogram stored o

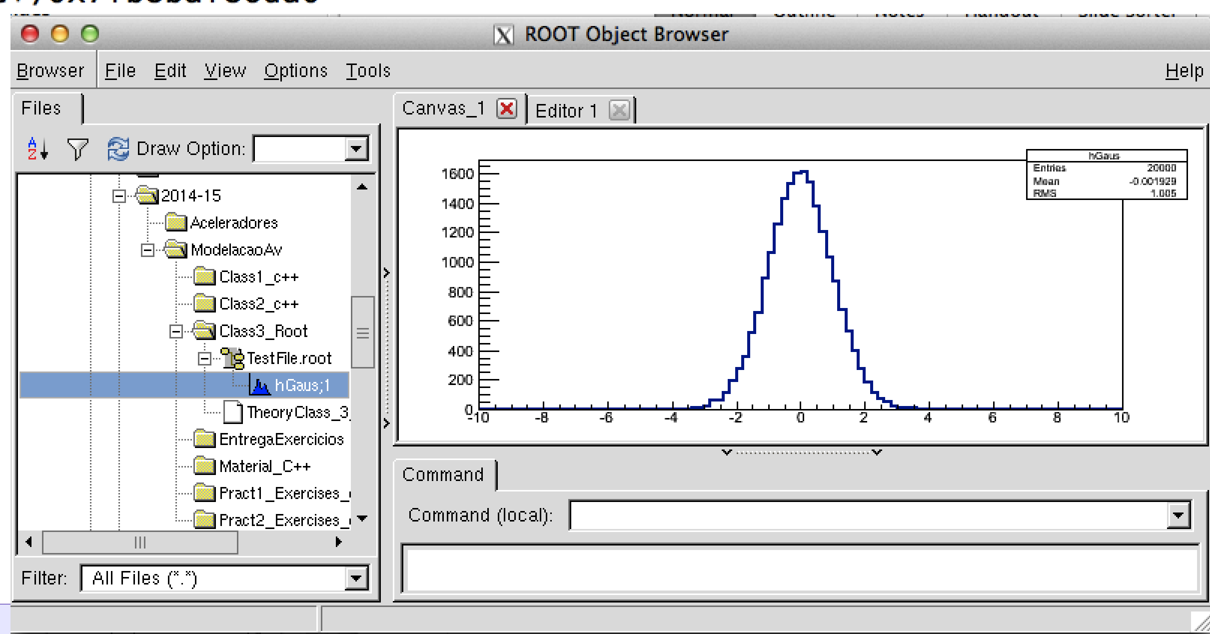
```
root [2] TH1F *h1 = (TH1F *) myFile.Get("hGaus")
root [3] h1->Draw()
Info in <TCanvas::MakeDefCanvas>:  created default
root [4]
```

- Draw the histogram.
- It creates a TCanvas for visualization

Using the GUI

- Create a Tbrowser object
- Navigate to the file, double click to open it, double click to draw the histogram

```
CINT/ROOT C/C++ Interpreter version 5.18.00, July 2, 2010
Type ? for help. Commands must be C++ statements.
Enclose multiple statements between { }.
root [0] TBrowser tb
root [1] (class TFile*)0x7fb8baf86da0
```



Useful member functions for files

- Options to open the file:

```
If option = NEW or CREATE   create a new file and open it for writing,
                            if the file already exists the file is
                            not opened.
          = RECREATE        create a new file, if the file already
                            exists it will be overwritten.
          = UPDATE          open an existing file for writing.
                            if no file exists, it is created.
          = READ            open an existing file for reading (default).
          = NET             used by derived remote file access
                            classes, not a user callable option
          = WEB             used by derived remote http access
                            class, not a user callable option
If option = "" (default), READ is assumed.
```

Useful member functions for files

- Enter the file:
  - File.cd()
- List the contents:
  - File.ls()
- Close:
  - File.Close()
- Get:
  - File.Get("name_of_object")
  - Returns a pointer of type TObject
  - Can be casted to the right type for further manipulation

- ROOT's own convention
  - Member functions start with capital letters
- cd, ls: copied from linux, use lower case letters

Using ROOT from a macro

```
void OpenFileExample(){


   std::cout << "ROOT program starting " << std::endl;


   // Opening the file
  TFile myFile("TestFile.root");

  //Retrieving the histogram:
  TH1F *h1 = (TH1F*) myFile->Get("hGaus");

  // Create a canvas to draw the histogram:
  TCanvas *myCanvas = TCanvas::MakeDefCanvas();

   // Drawing the histogram:
   h1->Draw();

  // Save canvas as a picture:
  myCanvas->SaveAs("outputHistogram.png");

 // Close file:
  myFile->Close();


  std::cout << "ROOT program finishing. Returning success " << std::endl;
}
```

Using ROOT from a macro

- Executing the macro from the CINT command line:

```
  *********************************************

ROOT 5.34/01 (tags/v5-34-01@45048, Jul 13 2012, 15:31:31 on macosx64)

CINT/ROOT C/C++ Interpreter version 5.18.00, July 2, 2010
Type ? for help. Commands must be C++ statements.
Enclose multiple statements between { }.
root [0] .L example1.cxx+
root [1] OpenFileExample()
ROOT program starting
Info in <TCanvas::Print>: png file outputHistogram.png has been created
ROOT program finishing. Returning success
root [2]
```

- Load and compile the macro
  - Can be loaded without compilation
    - .L example1.cxx
  - Compilation helps resolving problems that might end in malfunctioning

Using ROOT from a macro

- Executing the macro from the CINT command line:

```
  ********************************************

ROOT 5.34/01 (tags/v5-34-01@45048, Jul 13 2012, 15:31:31 on macosx64)

CINT/ROOT C/C++ Interpreter version 5.18.00, July 2, 2010
Type ? for help. Commands must be C++ statements.
Enclose multiple statements between { }.
root [0] .L example1.cxx+
root [1] OpenFileExample()
ROOT program starting
Info in <TCanvas::Print>: png file outputHistogram.png has been created
ROOT program finishing. Returning success
root [2]
```

- Execute the macro
  - Produced some printouts
  - Draws the histogram and saves the canvas in a png file
  - When closing the file, the histogram is removed from the canvas!

Using ROOT from a program

```
int main(){

   std::cout << "ROOT program starting " << std::endl;

   // Opening the file
  TFile myFile("TestFile.root");

  //Retrieving the histogram:
  TH1F *h1 = (TH1F*) myFile.Get("hGaus");

  // Create a canvas to draw the histogram:
  TCanvas myCanvas(true);

   // Drawing the histogram:
   h1->Draw();

  // Save canvas as a picture:
  myCanvas.SaveAs("outputHistogram.png");

 // Close file:
  myFile.Close();


  std::cout << "ROOT program finishing. Returning success " << std::endl;
}
```

Using ROOT from a program

- The makefile has to link with the ROOT libraries:
    - In this example linking with all possible root libraries
        - Less efficient but same makefile can be used for all possible programs

Examples — vim — 108×33

```
CC = g++
CFLAGS  = -W -Wall -fPIC
# Flags: -fPIC used to create a library that might be later on included in another library
#        -Wall: enables compiler's warning messages.
#
RCFLAGS = $(shell root-config --cflags)
RGLIBS  = $(shell root-config --glibs)

example : example.o
	$(CC) -o example $(CFLAGS) $(RCFLAGS) example.o $(RGLIBS)
example.o : example.cxx
	$(CC) $(CFLAGS) $(RCFLAGS) -c example.cxx

# make clean
clean :
	rm -rf example *.o *.d *.so
```

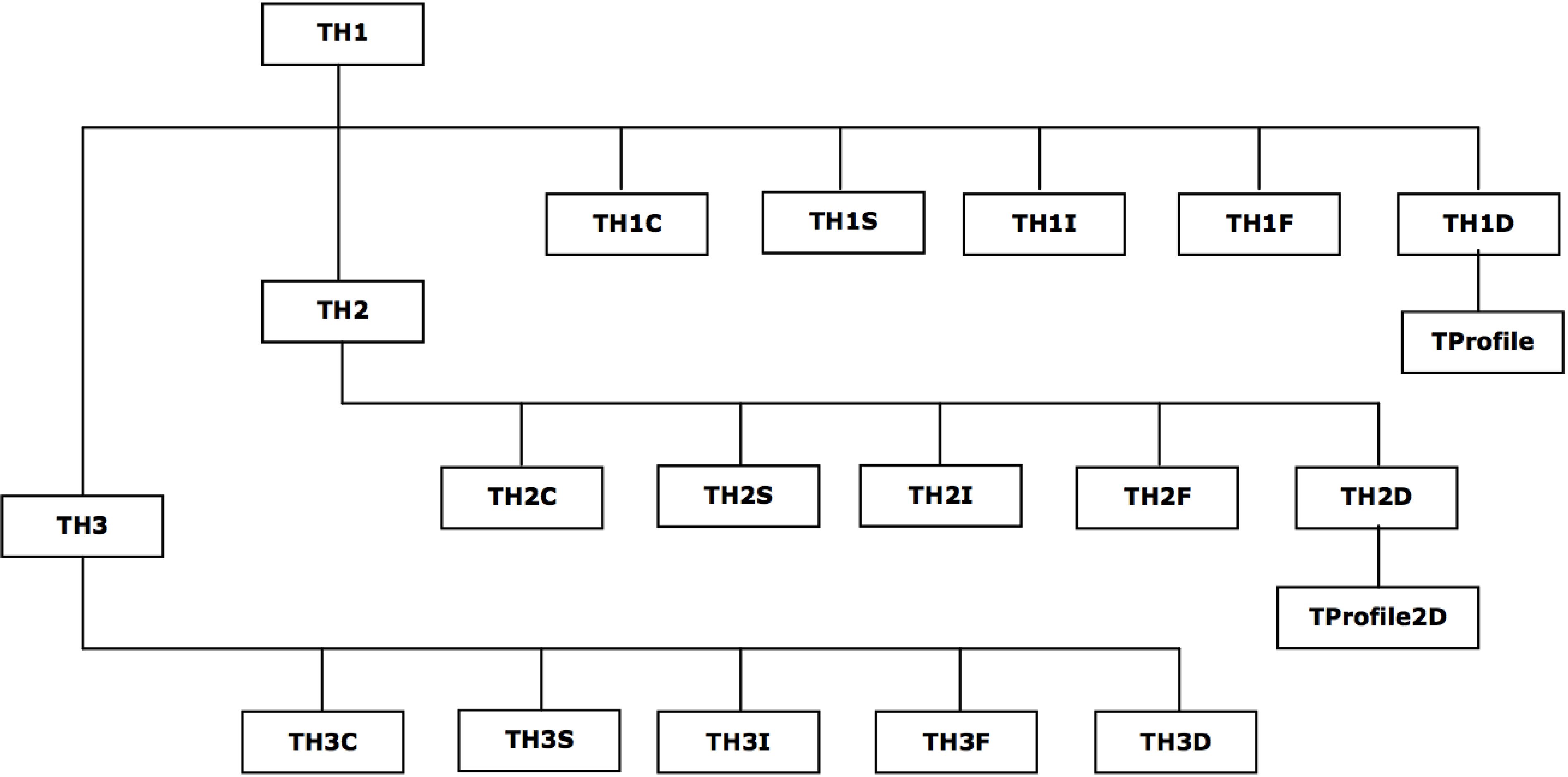
Notice

- When using the root interpreter or non-compiled macros:
  - No need to use includes
  - In many cases wrong syntax doesn't produce errors
    - may be interpreted correctly or not!
- When using compiled macros or programs
  - Need to use include files
  - Syntax errors during compilation
    - Safer! Reduce malfunctioning due to syntax errors
  - Faster execution
    - Important for long programs or running over long input data files

Notice

- When using the root interpreter or non-compiled macros:
  - No need to use includes
  - In many cases wrong syntax doesn't produce errors
    - may be interpreted correctly or not!
- When using compiled macros or programs
  - Need to use include files
  - Syntax errors during compilation
    - Safer! Reduce malfunctioning due to syntax errors
  - Faster execution
    - Important for long programs or running over long input data files

Histograms


TH1
TH1C
TH1S
TH1I
TH1F
TH1D
TProfile
TH2
TH2C
TH2S
TH2I
TH2F
TH2D
TProfile2D
TH3
TH3C
TH3S
TH3I
TH3F
TH3D

Histograms

Dimensions

- Histogram objects are called TH1F,TH1D, TH2F,TH3D

Float, Double

- To create a new histograms with 20 bins, in range [0,400] you have to do:

```
root[2] TH1F hist("hist","hist",20,0,400);
root[3] calls->Draw("time>>hist")
```

- Now we can do a lot of things on the histograms
  - Changing the properties, fitting, asking integrals, value of bins, overflow, uderflow, scaling, drawing normalized, etc…

Histograms

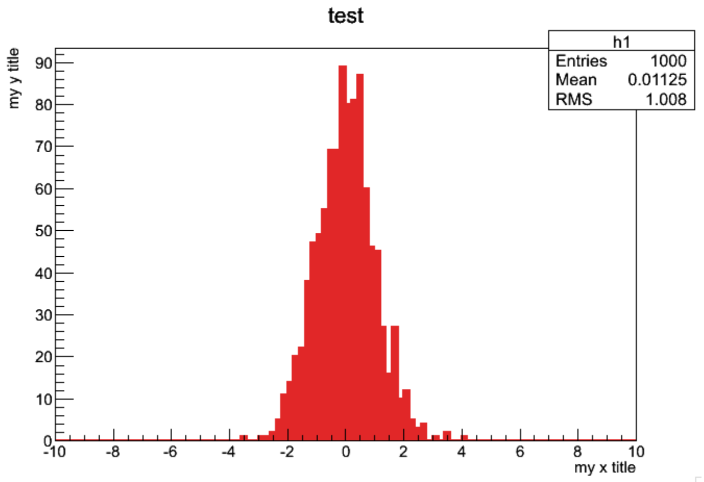
- Drawing:

  h->Draw("E");

- Many possible options (see manual):

"AXIS"	Draw only axis
"AH"	Draw histogram, but not the axis labels and tick marks
"]["	When this option is selected the first and last vertical lines of the histogram are not drawn.
"B"	Bar chart option
"C"	Draw a smooth Curve throught the histogram bins
"E"	Draw error bars
"E0"	Draw error bars including bins with o contents
"E1"	Draw error bars with perpendicular lines at the edges
"E2"	Draw error bars with rectangles
"E3"	Draw a fill area throught the end points of the vertical error bars
"E4"	Draw a smoothed filled area through the end points of the error bars
"L"	Draw a line throught the bin contents
"P"	Draw current marker at each bin except empty bins
"P0"	Draw current marker at each bin including empty bins

Useful histograms operations



- histogram->SetXTitle(“my x title”);
- histogram->SetYTitle(“my y title”);
- histogram->SetLineColor(kRed);
- histogram->SetLineWidth(2);
- histogram->SetFillColor(2);
- histogram->Fill(x): adds an entry on the bin corresponding to the value of x
- histo->Write(): stores the histogram in the current open file
- histo->GetEntries(): returns the number of entries
- histo->Fit(“gaus”): fits the histogram with a gaussian
- ...

Filling histograms

- A histogram can be also filled by calling `TH1F::Fill`

```
root [0] TH1F::Fill(

Int_t Fill(Double_t x)
Int_t Fill(Double_t x, Double_t w)
Int_t Fill(const char* name, Double_t w)
```

```
// Create a histogram:

TH1F *h1 = new TH1F("h1","",100,0.,10.);
h1->SetXTitle("x title");
h1->SetYTitle("Number of entries");
h1->SetLineColor(kBlue);

// Fill the histogram with integers
for (int i=0; i<10.; i++)
   h1->Fill(i);
```

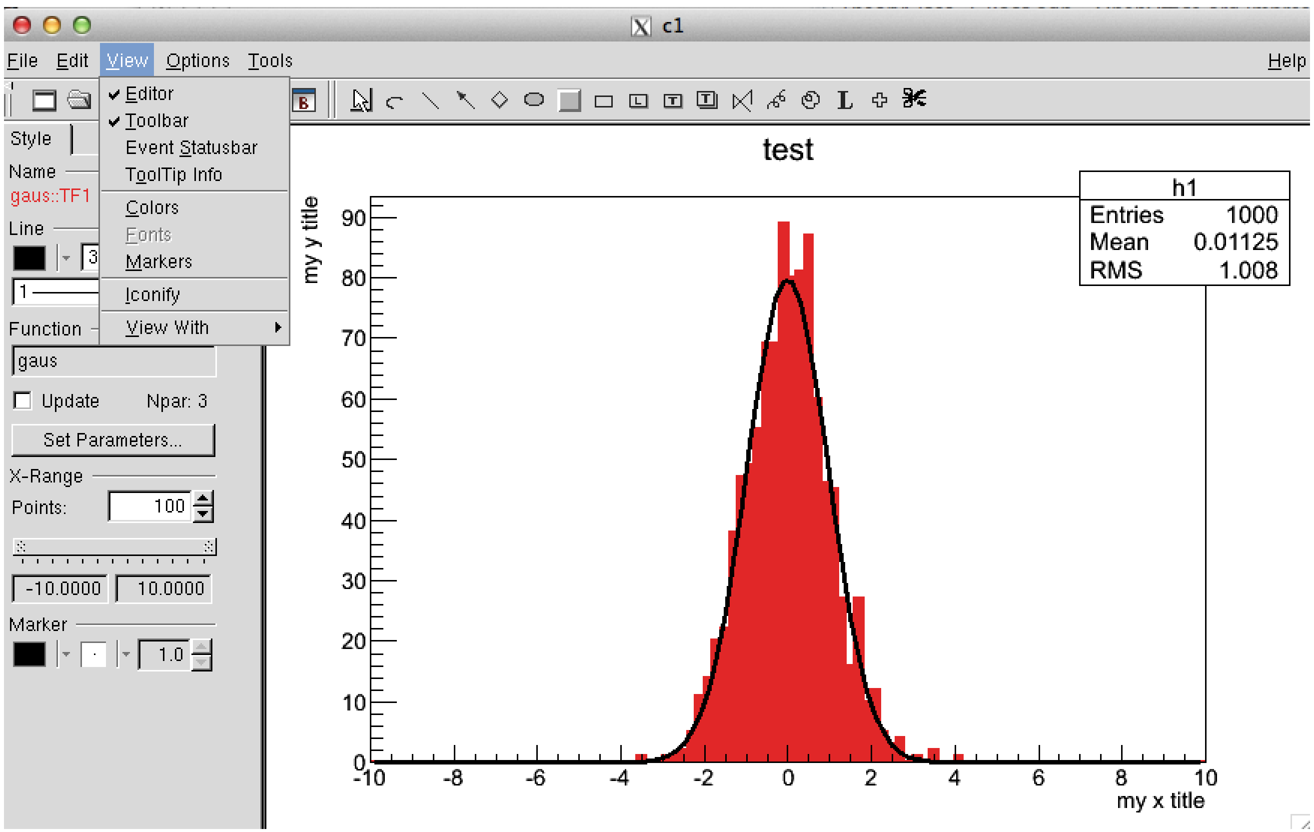
Print or save plots in different formats

void **SaveAs** (const char* filename = "", Option_t* option = "") const

```
Save Pad contents in a file in one of various formats.

  if filename is "", the file produced is padname.ps
  if filename starts with a dot, the padname is added in front
  if filename contains .eps, an Encapsulated Postscript file is produced
  if filename contains .pdf, a PDF file is produced
  if filename contains .svg, a SVG file is produced
  if filename contains .tex, a TeX file is produced
  if filename contains .gif, a GIF file is produced
  if filename contains .gif+NN, an  animated GIF file is produced
  if filename contains .xpm, a XPM file is produced
  if filename contains .png, a PNG file is produced
  if filename contains .jpg, a JPEG file is produced
```

Using the graphical editor



- Very easy to change the style
- Can be used to store a macro for later reference

  Fast to learn how to use different styles!

TTrees

- Class designed to store large quantities of same class objects
    - Optimized to reduce disk space
    - Enhanced access speed

  See example BasicTree