

粒子物理与核物理实验中的数据 分析

张黎明
清华大学

第三讲：ROOT在数据分析中的应用(1)

本讲要点

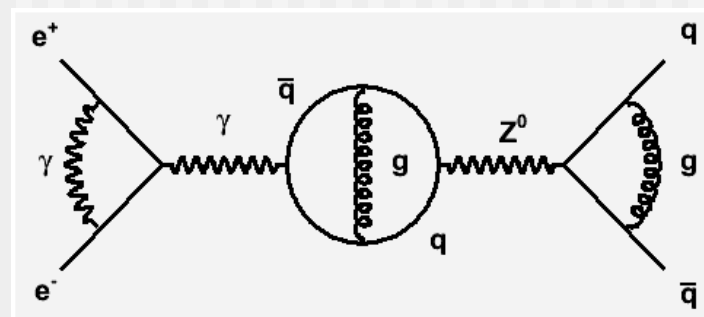
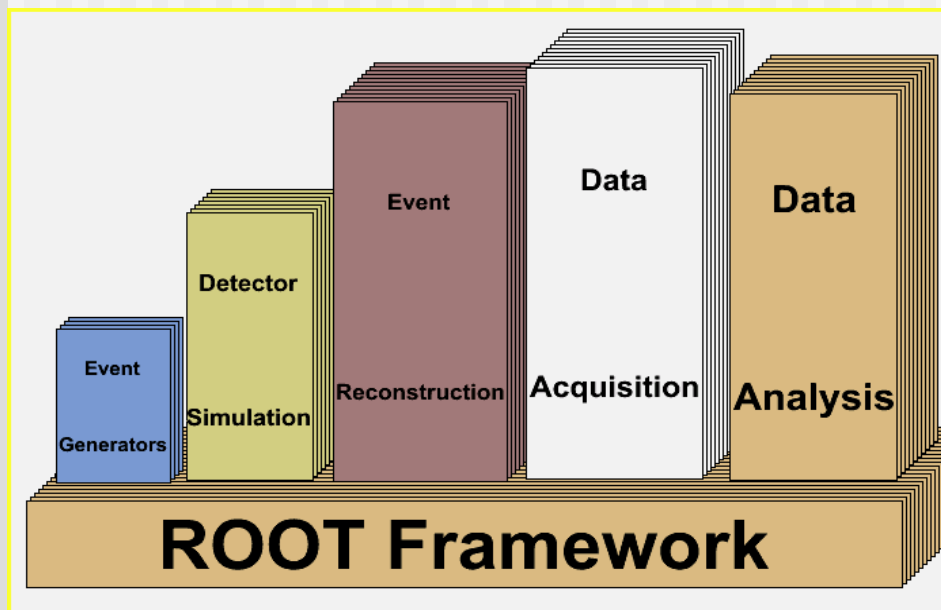
- 什么是ROOT ?
- ROOT体验中心
- 安装登录ROOT环境
- Tutorial 目录
- ROOT的常用指令
- ROOT的结构、语法简介
- ROOT的随机数
- ROOT的函数
- ROOT直方图

什么是 ROOT ?

ROOT: Executive Summary

... provides a set of **OO frameworks** with **all the functionality** needed to handle and analyse **large amounts of data** in a **very efficient** way....

关键字：面向对象的框架、所有功能、海量数据、非常有效



结论：很不谦虚！

ROOT简介

跟数据分析有关的东西，基本都是ROOT的擅长：

- 数据处理、数据分析、数据可视化、数据存储

ROOT主要是C++语言写成的（C++ 11标准）

- 提供Python绑定



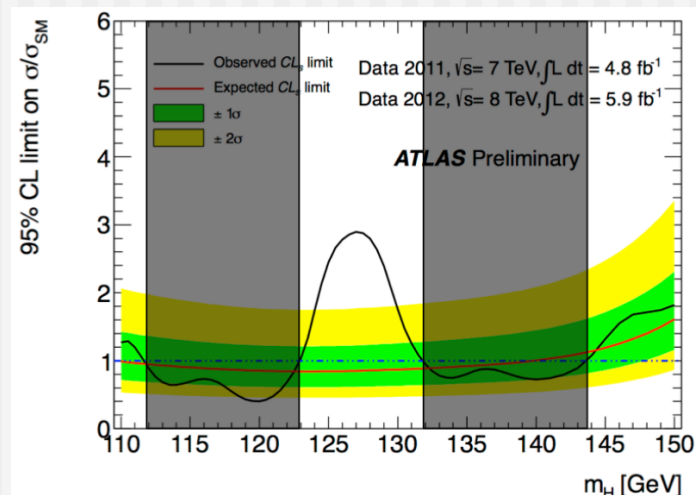
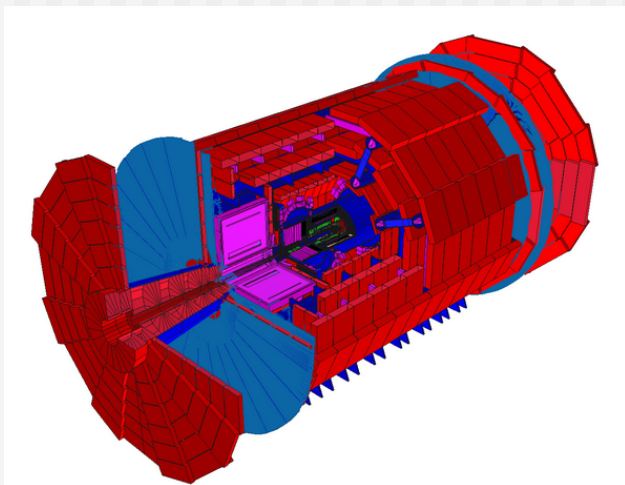
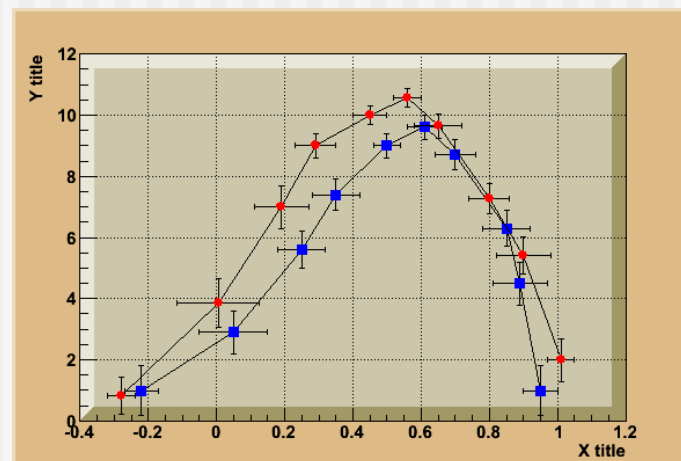
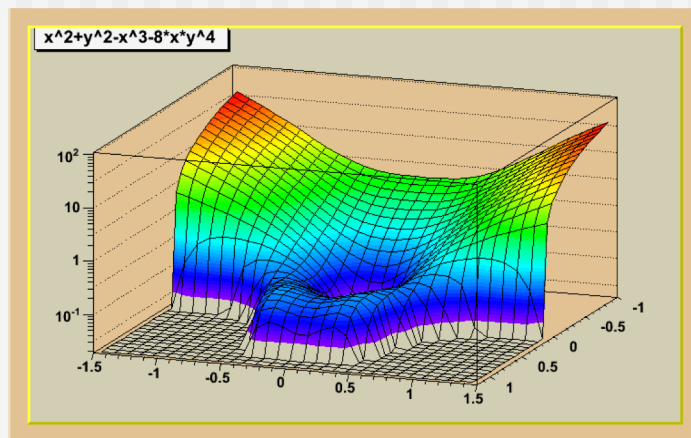
ROOT包括：

- 数据分析：histograms, graphs, trees
- I/O: 按行或按列的C++对象的存储
- 统计工具（RooFit/RooStats）
- Math：复杂的数学函数（如Erf, Bessel）
- Multivariate Analysis (TMVA)
- 更多：HTTP severing, JavaScript可视化, event display
- PROOF：parallel analysis facility

ROOT体验中心

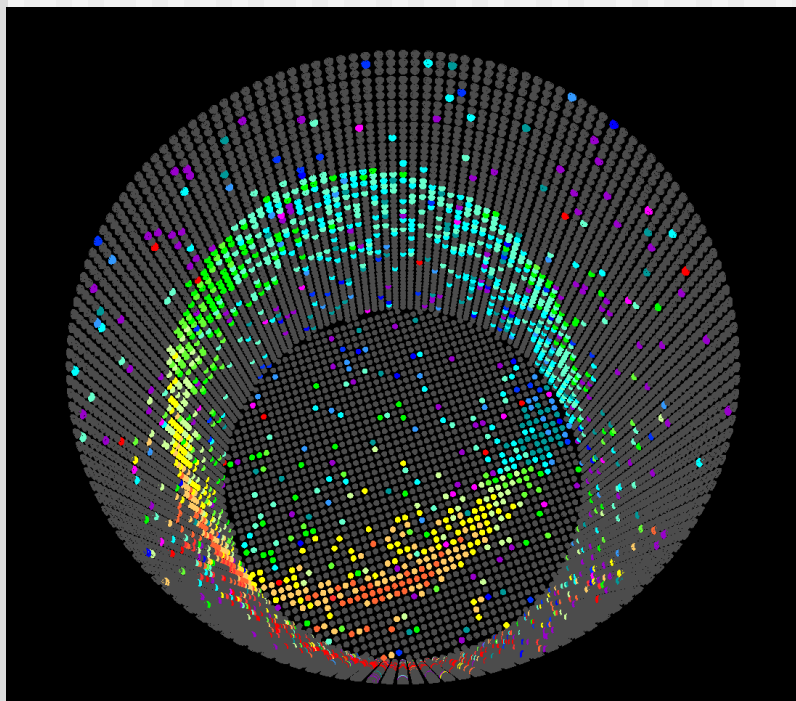
还可以在ROOT网站上看到一些ROOT图片：

<https://root.cern.ch/gallery>

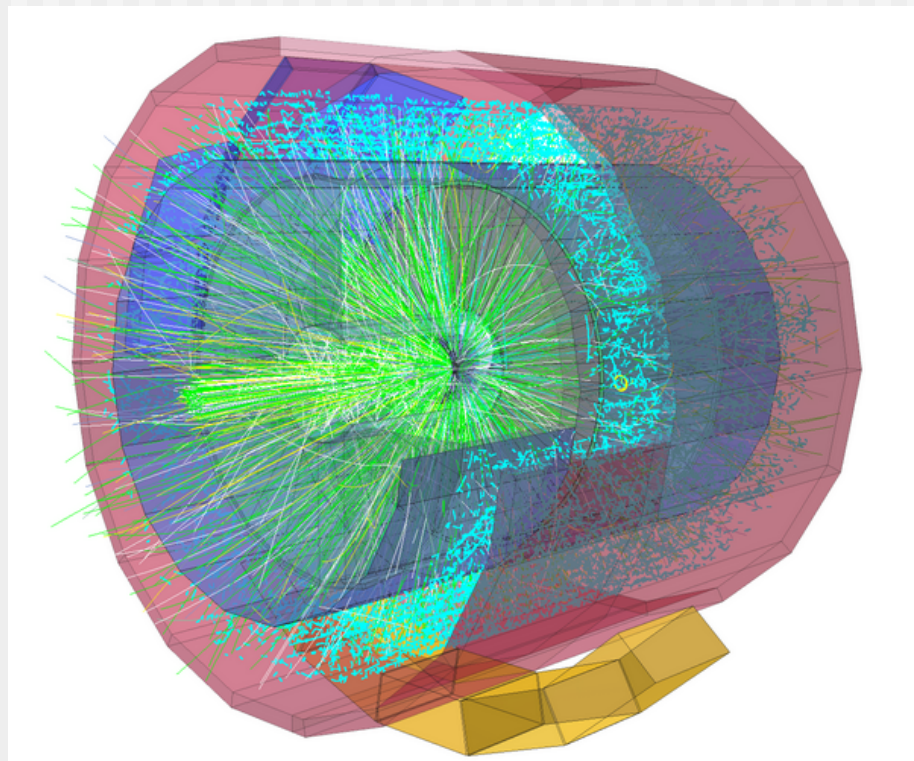


事例显示

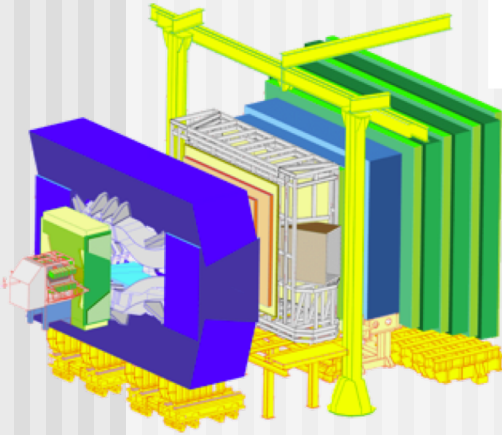
超级神冈重建事例



Alice 模拟重建事例



LHCb实验的数据处理



Data

Event Filtering

Offline Processing

Root

Analysis

Reconstruction

Stripping,
Further
processing

Event
Selection,
statistical
treatment ...

Raw

Reco

...

Analysis
Formats

Images

Root-like format

Root file

安装ROOT

到ROOT主页下载需要的版本到指定目录。

(<http://root.cern.ch/drupal/content/downloading-root>)

下载源代码:

<https://root.cern.ch/downloading-root>

由源代码安装: (基本过程, 订制安装可能需要一些参数)

`tar xvfz *.tar.gz`, 假定解压后的文件在root目录中,

`> cd root`

`> ./configure [--prefix=/home/zhanglm/root (安装目录)]`

`> make`

(自行安装的时候有很多麻烦, 主要是缺少一些依赖的软件, 你的linux系统总包含一些软件库、管理器之类, 很容易得到解决。)

使用:

`source bin/thisroot.sh (或者 .csh)`

(这个还可以加在你的登录环境中.bashrc或.cshrc, 自动加载)

`root`

(进入root)

登录ROOT环境

■ 运行 `>root`

■ 退出 `root[0] .q`

■ 键入 `help` 指令，如

`root[0].?`

`root[1].ls`

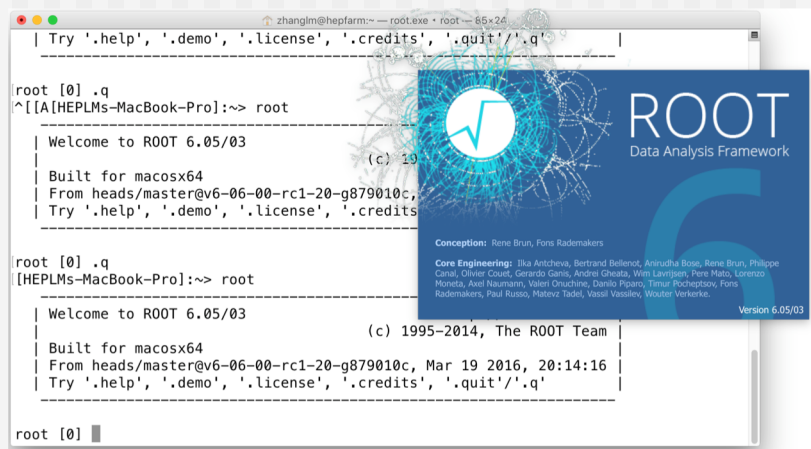
`root[2].!ls`

其他：

• `.ls` 显示ROOT当前环境的所有信息

• `!.ls` 显示Linux系统当前目录的所有信息

• 注：ROOT环境中，ROOT指令都以“.”开头，系统指令都以“!”开头



```
zhanglm@heplarm: ~ - root.exe - root - 65x24
| Try '.help', '.demo', '.license', '.credits', '.quit'/'.'q |
|
root [0] .q
^[[A[HEPLMs-MacBook-Pro]:~> root
-----
| Welcome to ROOT 6.05/03
| (c) 1995-2014, The ROOT Team
| Built for macosx64
| From heads/master@v6-06-00-rc1-20-g879010c,
| Try '.help', '.demo', '.license', '.credits'
|
root [0] .q
[HEPLMs-MacBook-Pro]:~> root
-----
| Welcome to ROOT 6.05/03
| (c) 1995-2014, The ROOT Team
| Built for macosx64
| From heads/master@v6-06-00-rc1-20-g879010c, Mar 19 2016, 20:14:16
| Try '.help', '.demo', '.license', '.credits', '.quit'/'.'q'
|
root [0] █
```

各种执行方式

课上演示, hsimple.C

```
1. > root hsimple.C
```

```
2. > root
```

```
    root [0] .x hsimple.C
```

```
3. > root
```

```
    root [0] .L hsimple.C
```

```
    root [1] hsimple()
```

```
4. > root
```

```
    root [0] .L hsimple.C+
```

```
    root [1] hsimple()
```

解释执行

编译执行

编译执行会执行最严格的语法检查，
程序运行最安全，推荐使用

执行命令行选项

```
% root -?
```

```
Usage: root [-l] [-b] [-n] [-q] [dir] [[file:]data.root]
                                         [file1.C ... fileN.C]
```

Options:

- b : run in batch mode without graphics
- n : do not execute logon and logoff macros as specified in .rootrc
- q : exit after processing command line macro files
- l : do not show splash screen
- x : exit on exception

dir : if dir is a valid directory cd to it before executing

- ? : print usage
- h : print usage
- help : print usage
- config : print ./configure options
- memstat : run with memory usage monitoring

另一种执行方式

执行一个C脚本：在本底运行 (-b)、把输出存成一个文件、运行完退出ROOT (-q)

For example if you would like to run a script `myMacro.C` in the background, redirect the output into a file `myMacro.log`, and exit after the script execution, use the following syntax:

```
root -b -q myMacro.C > myMacro.log
```

If you need to pass a parameter to the script use:

```
root -b -q 'myMacro.C(3)' > myMacro.log
```

Be mindful of the quotes, i.e. if you need to pass a string as a parameter, the syntax is:

```
root -b -q 'myMacro.C("text")' > myMacro.log
```

You can build a shared library with ACLiC and then use this shared library on the command line for a quicker execution (i.e. the compiled speed rather than the interpreted speed). See also “Cling the C++ Interpreter”.

```
root -b -q myMacro.so > myMacro.log
```


Tutorial 目录

课上演示

在\$ROOTSYS/tutorials目录下，有五花八门的例子。
以后会经常与这个目录打交道。先尝试一下吧。

尝试方法：

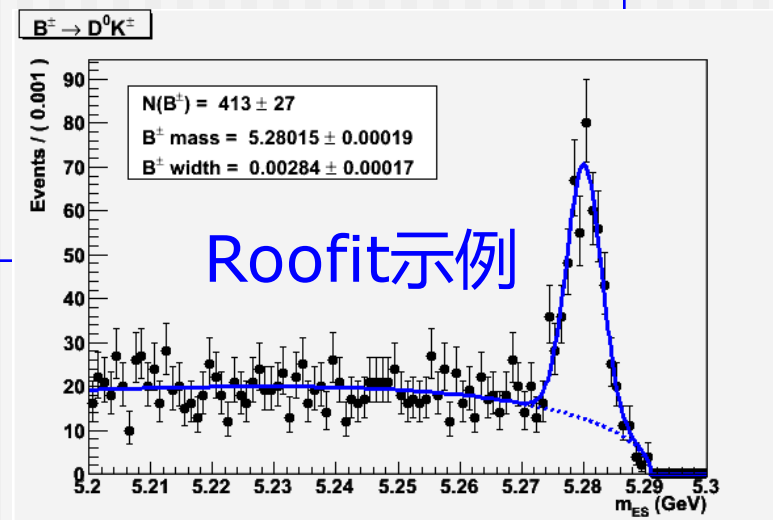
```
>cd $HOME
```

```
>cp -r $ROOTSYS/tutorials . (注意不要把这个"."漏掉了)
```

```
>cd tutorials
```

然后找个感兴趣的目录/文件，
执行ROOT脚本，比如

```
>root -l demos.C
```



小技巧提示：

根据关键字"xxxx"从tutorials的例子中寻找线索

```
grep -sirn "xxxx" $ROOTSYS/tutorials
```

比如找随机数用法：

```
grep -sirn "random" $ROOTSYS/tutorials
```

ROOT语法—基本信息

- ROOT使用C++语法

一段C++程序可以直接在ROOT环境运行

- 数据类型重定义

int → Int_t

float → Float_t

double → Double_t

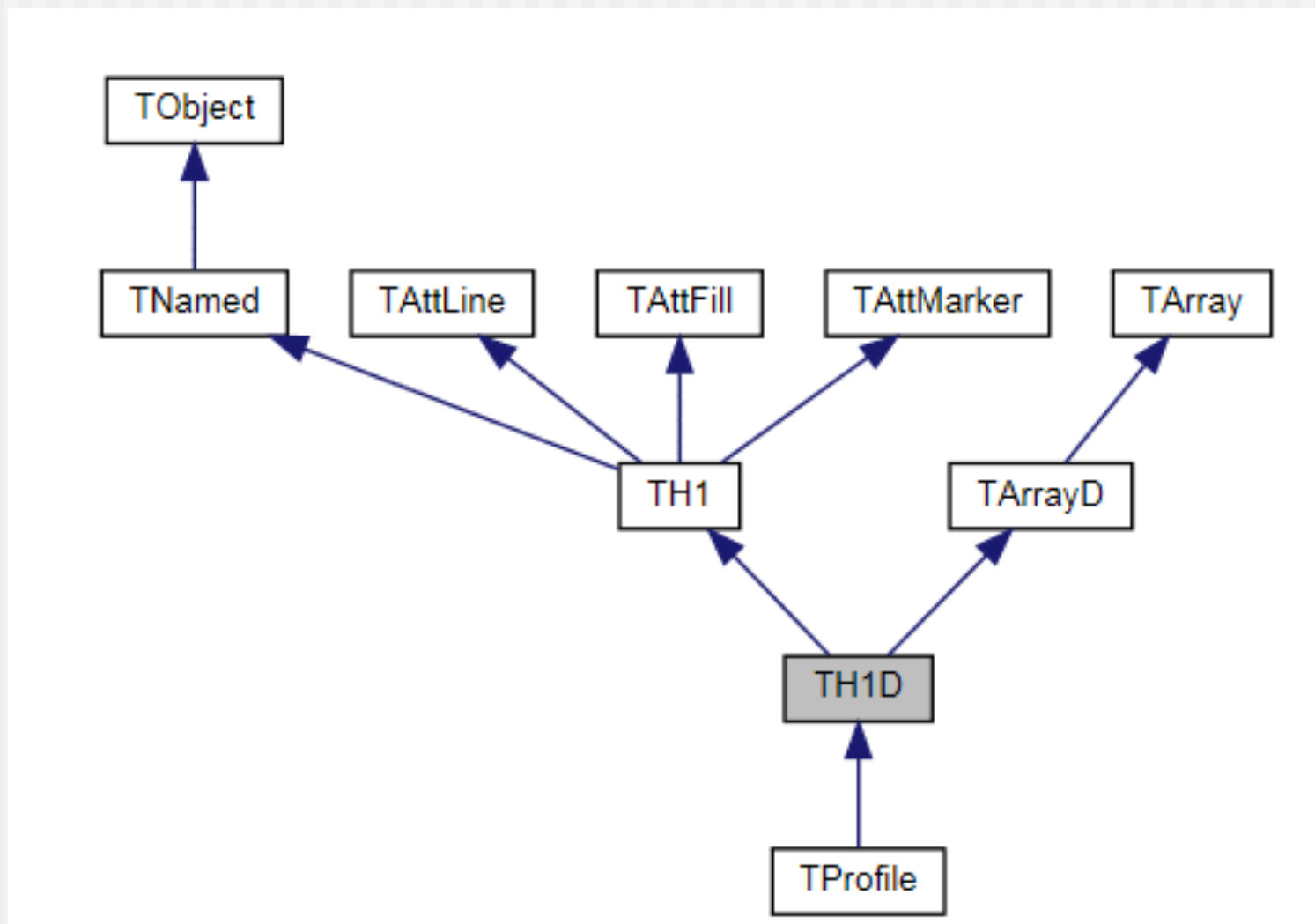
.....

- ROOT的类都以T开头

如TFile, TH1F, TTree, ...

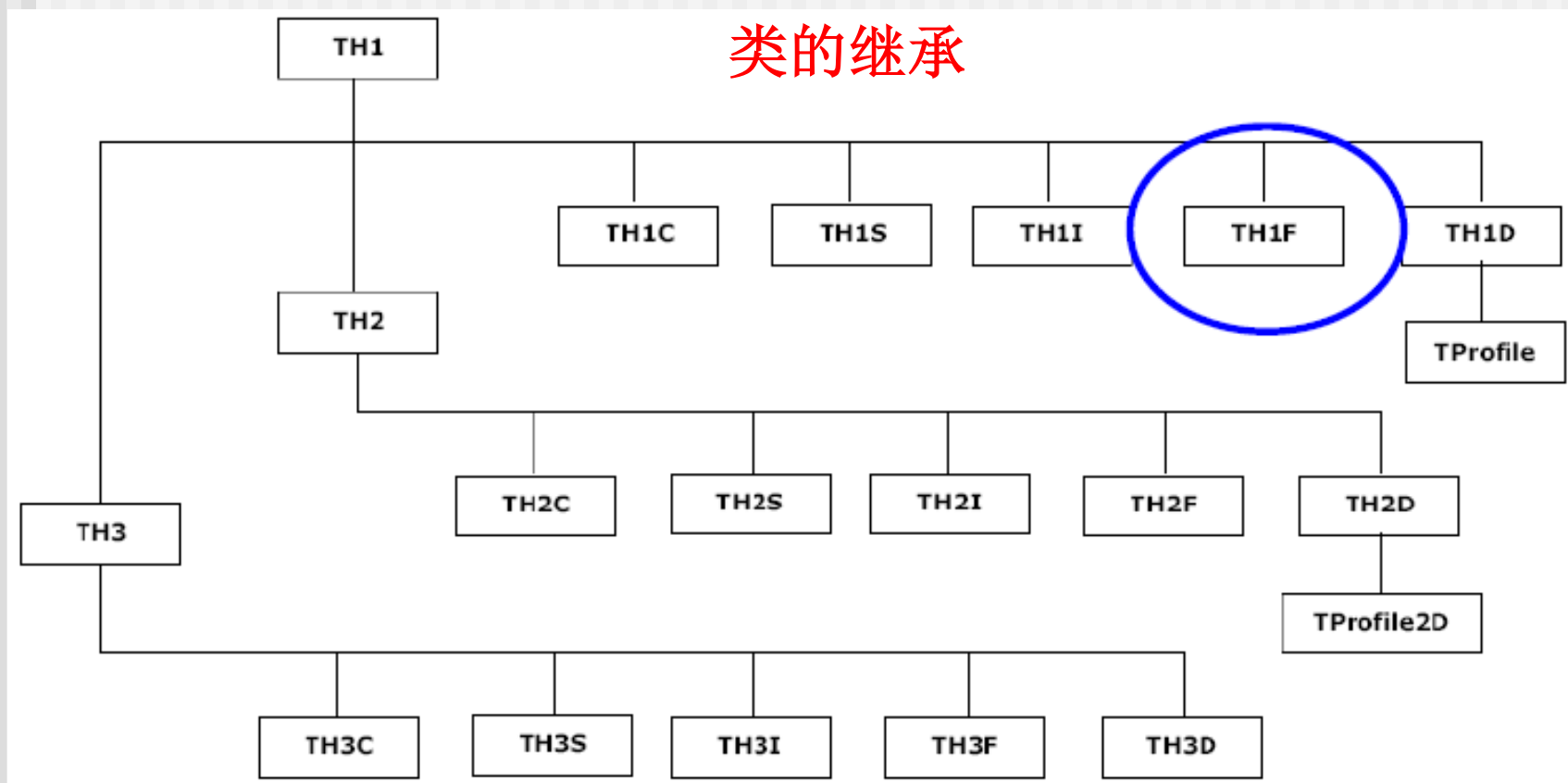
- 可以直接在ROOT环境中运行macro文件(自动调用cint编译器), 也可以在Makefile中设置好相关参数用g++编译得到可执行文件运行。

ROOT的关于继承的例子



ROOT结构一类的的使用

ROOT中有众多已经定义好的类可供使用，
比如直方图家族



ROOT的一些重要的类

其它常用类

基础类: TObject

数学函数: TF1, TF2, TF3...

图 形: TGraph, TGraphErrors, TGraph2D,...

文 件: TFile

画 布: TCanvas, TPad, ...

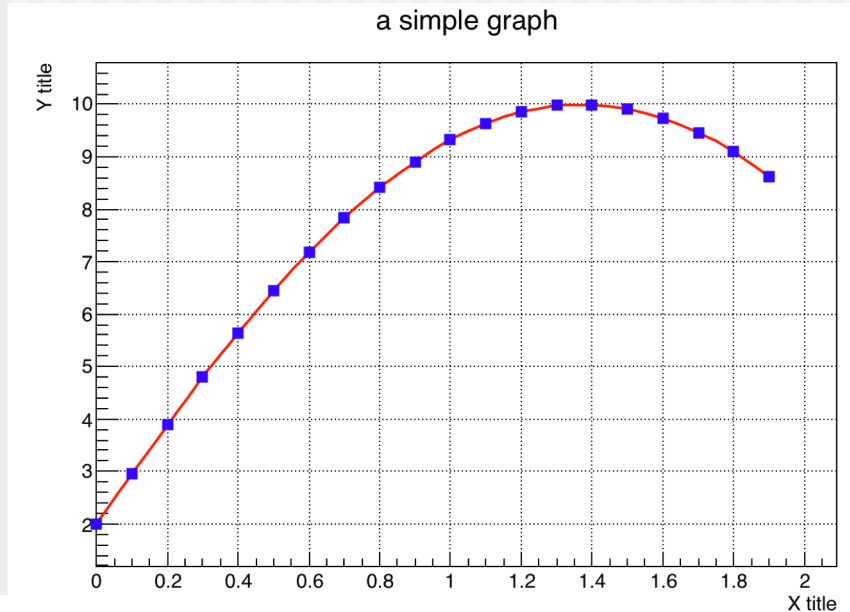
随 机 数: TRandom, TRandom1, TRandom2, TRandom3

树, 树链: TTree, TChain (非常重要)

还有很多全局函数, 全局类, 全局变量, 多数以g开头, 如:
gRandom, gROOT, gStyle, gPad, gEnv, gFile...

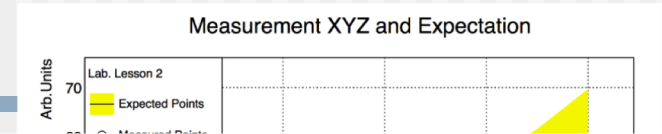
TGraph

- 图形
- Tab 查找 constructors



```
root [2] TGraph f(  
  TGraph TGraph()  
  TGraph TGraph(Int_t n)  
  TGraph TGraph(Int_t n, const Double_t* x, const Double_t* y) ←  
  TGraph TGraph(Int_t n, const Float_t* x, const Float_t* y)  
  TGraph TGraph(Int_t n, const Int_t* x, const Int_t* y)  
  TGraph TGraph(const TF1* f, Option_t* option = "")  
  TGraph TGraph(const TGraph& gr)  
  TGraph TGraph(const TH1* h)  
  TGraph TGraph(const TVectorD& vx, const TVectorD& vy)  
  TGraph TGraph(const TVectorF& vx, const TVectorF& vy)  
  TGraph TGraph(const char* filename, const char* format = "%lg %lg", Option_t* option = "") ←
```

读取ASCII文件



- Create a graph (TGraph)
- Set its title to “My graph”, its X axis title to “myX” and Y axis title to “myY”
- Fill it with three points: (1,0), (2,3), (3,4)
- Set a red full square marker
- Draw a orange line between points

```
TGraphErrors graph_expected("./macro2_input_expected.txt", "%lg %lg %lg");  
graph_expected.SetTitle("Measurement XYZ and Expectation;length [cm];Arb.Units");  
graph_expected.SetFillColor(kYellow);  
graph_expected.DrawClone("E3AL"); // E3 draws the band
```

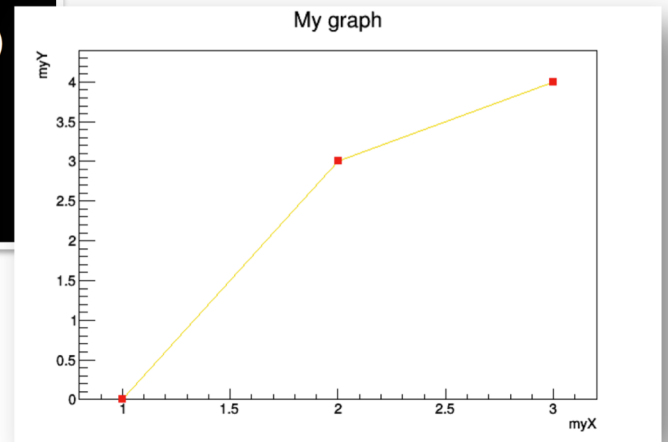
```
TGraphErrors graph("./macro2_input.txt", "%lg %lg %lg");  
graph.SetMarkerStyle(kCircle);  
graph.SetFillColor(0);  
graph.DrawClone("PESame");
```

```
TLegend leg(.1,.7,.3,.9,"Lab. Lesson 2");  
leg.SetFillColor(0);  
leg.AddEntry(&graph_expected,"Expected Points");  
leg.AddEntry(&graph,"Measured Points");  
leg.DrawClone("Same");
```

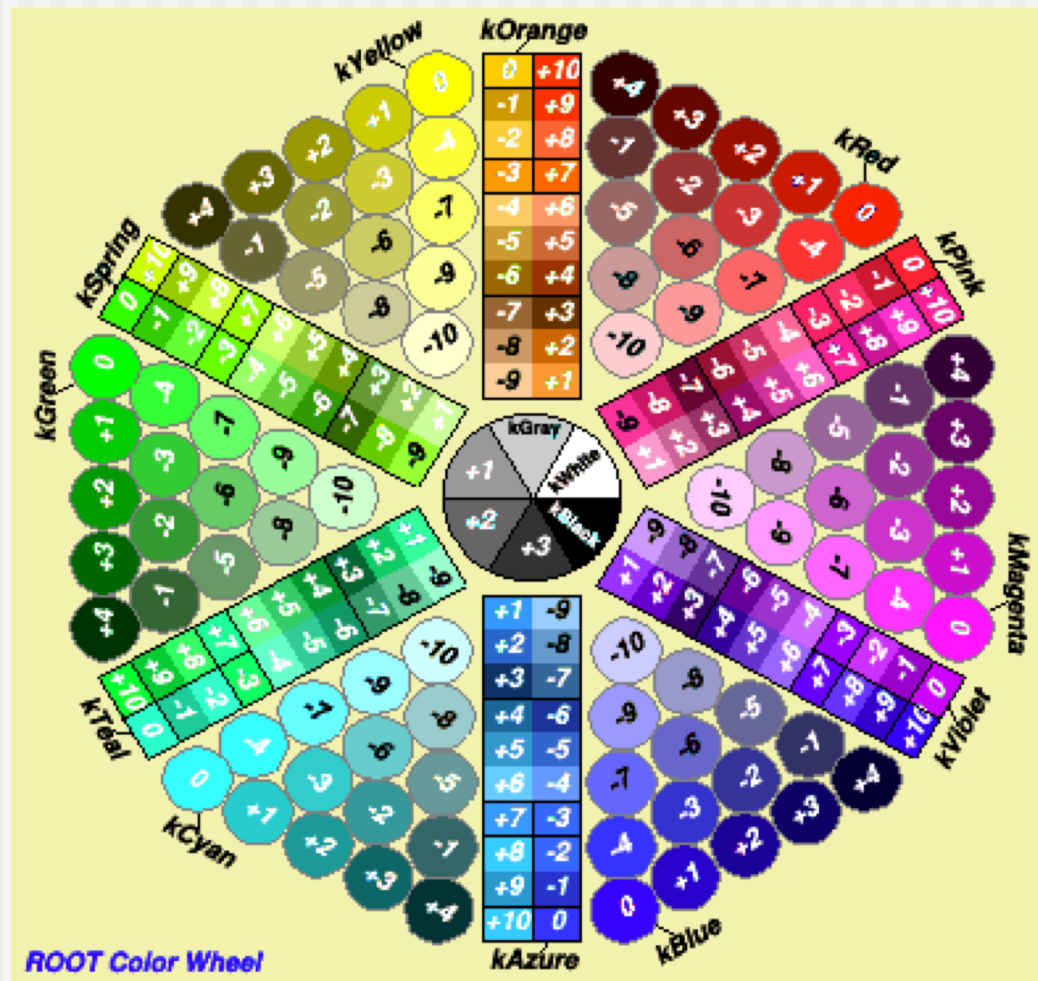
课堂作业

- Create a graph (TGraph)
- Set its title to “My graph”, its X axis title to “myX” and Y axis title to “myY”
- Fill it with three points: (1,0), (2,3), (3,4)
- Set a red full square marker
- Draw a orange line between points

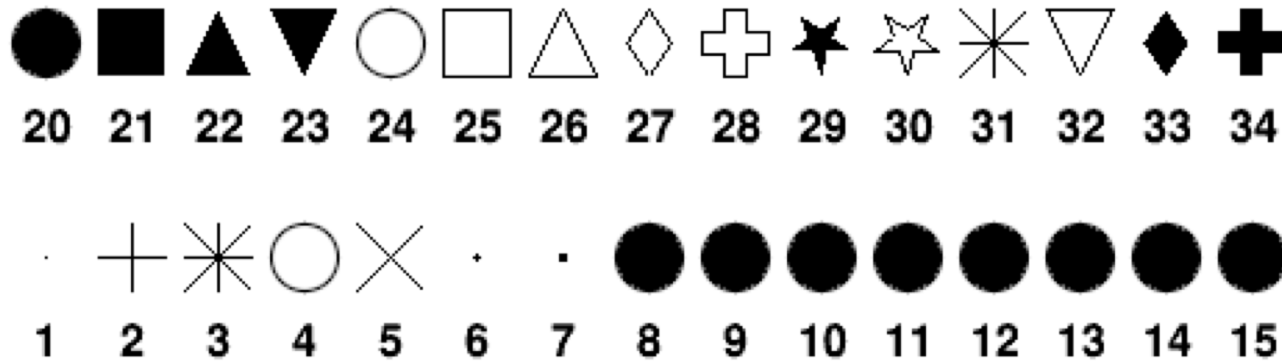
```
root [0] TGraph g
root [1] g.SetTitle("My graph;myX;myY")
root [2] g.SetPoint(0,1,0)
root [3] g.SetPoint(1,2,3)
root [4] g.SetPoint(2,3,4)
root [5] g.SetMarkerStyle(kFullSquare)
root [6] g.SetMarkerColor(kRed)
root [7] g.SetLineColor(kOrange)
root [8] g.Draw("APL")
```



Color



Marker



```
kDot=1, kPlus, kStar, kCircle=4, kMultiply=5,  
kFullDotSmall=6, kFullDotMedium=7, kFullDotLarge=8,  
kFullCircle=20, kFullSquare=21, kFullTriangleUp=22,  
kFullTriangleDown=23, kOpenCircle=24, kOpenSquare=25,  
kOpenTriangleUp=26, kOpenDiamond=27, kOpenCross=28,  
kFullStar=29, kOpenStar=30, kOpenTriangleDown=32,  
kFullDiamond=33, kFullCross=34
```

Also available
through more
friendly names 😊

上讲回顾和本节要点

- ROOT 基本概念 (C++, 实验数据处理)

- 安装与登录ROOT以及体验

- ROOT的语法简介

完全兼容C++语法, 解析运行

- 接下来讲

数学函数, 直方图, 随机数, 文件等

TMATH, TF1, TF2, TF3, TFile, TH1I,
TH1F, TH1D, TH2F, gRandom

ROOT的随机数

随 机 数: TRandom, TRandom1, TRandom2, TRandom3

周期	10^9	10^{171}	10^{26}	10^{6000}
速度(ns/call)	34	242	37	45

速度与CPU和编译器有关

gRandom是指向当前随机数产生子的指针,

该产生子默认**TRandom3**对象。

https://root.cern.ch/doc/v618/TRandom_8h.html

(为什么看TRandom? 因为TRandom1/2/3都继承自TRandom)

TRandom3 *rnd = new TRandom3(); ← **Default seed=4357**

TRandom3 *rnd = new TRandom3(0); ← **Seed根据global time, 永远不同**

ROOT—随机数

<code>gRandom->Binomial(ntot, p):</code>	二项分布
<code>gRandom->BreiWigner(mean, gamma)</code>	Breit-Wigner分布
<code>gRandom->Exp(tau)</code>	指数分布
<code>gRandom->Gaus(mean,sigma)</code>	高斯分布
<code>gRandom->Integer(imax)</code>	[0,imax-1]随机整数
<code>gRandom->Landau(mean,sigma)</code>	Landau分布
<code>gRandom->Poisson(mean)</code>	泊松分布(返回int)
<code>gRandom->PoissonD(mean)</code>	泊松分布(返回double)
<code>gRandom->Rndm()</code>	(0,1]均匀分布
<code>gRandom->Uniform(x1,x2)</code>	(x1,x2]均匀分布
....	

ROOT中统计直方图

定制一维直方图

```
TH1F *hist_name = new TH1F("hist_name","hist_title",  
num_bins,x_low,x_high);
```

定制二维图

```
TH2F *hist_name = new TH2F("hist_name","hist_title",  
num_bins_x,x_low,x_high,num_bins_y,y_low,y_high);
```

定制三维图

```
TH3F *hist_name = new TH3F("hist_name","hist_title",  
num_bins_x,x_low,x_high,num_bins_y,y_low,y_high,  
num_bins_z,z_low,z_high);
```

填充统计图

```
hist_name->Fill(x);  
hist_name->Fill(x,y);  
Hist_name->Fill(x,y,z);
```

绘图：

```
root[0]hist_name->Draw();
```

ROOT脚本文件示例:直方图, 随机数

```
void histw() {  
    gROOT->Reset();  
    const Int_t NEntry = 10000 ;  
    TFile *file = new TFile("Landau.root","RECREATE");  
    TH1F *h1 = new TH1F("h1","A simple histo",100,0,1);  
    for (int i=0;i<NEntry;i++) {  
        h1->Fill( gRandom->Landau(0.2,0.1) );  
    }  
    h1->GetXaxis()->SetTitle("X Title,  $\sqrt{\Delta_2}$ ");  
    h1->GetXaxis()->CenterTitle();  
    h1->GetYaxis()->SetTitle("Y Title, Entries/0.01");  
    TCanvas *c1 = new TCanvas("c1","");  
    h1->Draw();  
    file->Write();  
    c1->SaveAs("Landau.pdf");  
}
```

1. 文件准备, 写入

2. 生成直方图

3. 填图

4. 生成图片

打开ROOT文件

1. 打开已有的root文件，如刚刚生成的Landau.root:
终端提示行下:

```
> root -l Landau.root
```

2. ROOT环境下:

```
> root
```

```
root [0] TFile f1("Landau.root");
```

```
root [1] .ls
```

```
root [2] h1->Draw();
```

3. 利用TBrowser



```
> root
```

```
root [0] TBrowser b
```

```
利用你的鼠标 .....
```

4. 当然也可以利用C++代码执行

打开一个文件，查看一个直方图

```
//  
//  open a root file and get a histogram  
//  
void histr() {  
  
    TFile *file = new TFile("Landau.root", "READ");  
    TH1F *h1 = (TH1F*) file->Get("h1");  
    h1->Draw();  
}
```

1. 文件打开

2. 得到直方图的指针

3. 画图

很多情况下，这些命令可以在**root**的提示符下逐行键入，并进行测试。

ROOT中的数学函数

画图时采用

```
root[1]fun_name.Draw();
```

□制作一维函数曲线图

```
TF1 *fun_name = new TF1("fun_name","expression",  
x_low,x_high);
```

```
root[0]TF1 *f1 = new TF1("f1","x*sin(x)",-5,5);
```

□制作二维函数曲线图

```
TF2 *fun_name = new TF2("fun_name","expression",  
x_low,x_high, y_low,y_high);
```

```
root[0]TF2 *f2 = new TF2("f2","x*sin(x)+y*cos(y)",  
-5,5,-10,10);
```

□制作三维函数曲线图

```
TF3 *fun_name = new TF3("fun_name","expression",  
x_low,x_high,y_low,y_high,z_low,z_high);
```

```
root[0]TF3 *f3 = new TF3("f3","x*sin(x)+y*cos(y)  
+z*exp(z)",-5,5,-10,10,-20,20);
```

数学函数的定义方式(1)

ROOT中定义数学函数的方式多种多样

❑ 利用C++数学表达式

```
TF1* f1 = new TF1("f1", "sin(x)/x", 0, 10);
```

❑ 利用TMath定义的函数

```
TF1 *f1 = new TF1("f1", "TMath::DiLog(x)", 0, 10);
```

❑ 利用自定义C++数学函数

```
Double_t myFun(double x) {  
    return x+sqrt(x);  
}  
TF1* f1 = new TF1("f1", "myFun(x)", 0, 10);
```

以上函数都不含参数，但在数据拟合时，我们往往需要定义含未知参数的函数

数学函数的定义方式(2)

ROOT中定义含未知参数的数学函数

□ ROOT已经预定义了几种常用的含参函数

gaus: 3个参数

$$f(x) = p0 * \exp(-0.5 * ((x - p1) / p2)^2)$$

expo: 2个参数

$$f(x) = \exp(p0 + p1 * x)$$

polN: N+1个参数

$$f(x) = p0 + p1 * x + p2 * x^2 + \dots$$

其中 $N=0, 1, 2, \dots$ ，使用时根据需要用 `pol0, pol1, pol2...`

landau: 3个参数

朗道分布，没有解析表达式

这些预定义函数可直接使用，比如

`histogram->Fit("gaus");` // 对直方图进行高斯拟合

`TF1 *f1=new TF1("f1","gaus",-5,5);`

数学函数的定义方式(3)

ROOT中自定义含未知参数的数学函数

❑ 利用C++数学表达式

```
TF1* f1 = new TF1("f1", "[0]*sin([1]*x)/x", 0, 10);  
f1->SetParameters(2., 1.5);
```

❑ 利用C++数学表达式以及ROOT预定义函数

```
TF1* f1 = new TF1("f1", "gaus(0)+[3]*x", 0, 3);
```

❑ 利用自定义的C++数学函数

```
Double_t myFun(Double_t *x, Double_t *par) {  
    Double_t f=par[0]*exp(-x[0]/par[1]);  
    return f;  
}
```

```
TF1* f1 = new TF1("f1", myFun, 0, 10, 2);  
f1->SetParameter(0, 100);  
f1->SetParameter(1, 2);
```

指定参数数目



ROOT脚本文件示例: Macro文件



```
// Macro myfunc.C
```

```
Double_t myfunction(Double_t *x, Double_t *par)
```

```
{
```

```
    Float_t xx =x[0];
```

```
    Double_t f = TMath::Abs(par[0]*sin(par[1]*xx)/xx);
```

```
    return f;
```

```
}
```

```
void myfunc()
```

```
{
```

```
    TF1 *f1 = new TF1("myfunc",myfunction,0,10,2);
```

```
    f1->SetParameters(2,1);
```

```
    f1->SetParNames("constant","coefficient");
```

```
    f1->Draw();
```

```
}
```

```
void myfit()
```

```
{
```

```
    TH1F *h1=new TH1F("h1","test",100,0,10);
```

```
    h1->FillRandom("myfunc",20000);
```

```
    TF1 *f1 = (TF1 *)gROOT->GetFunction("myfunc");
```

```
    f1->SetParameters(800,1);
```

```
    h1->Fit("myfunc");
```

```
}
```

← 解释执行的条件下, 要与文件名同名

```
Root > .L myfunc.C
Root > myfunc();
Root > myfit();
```

作业

1. 自己生成两维的直方图，利用随机数生成两维的高斯分布，存成文件，然后自己用两种以上的方法打开，重新查看。
2. 写一个两维高斯函数拟合上面产生的直方图

参考资料

- ROOT手册第2章，第3章
- <http://root.cern.ch>
- <http://root.cern.ch/root/Reference.html>
- <http://root.cern.ch/root/Tutorials.html>
- <http://root.cern.ch/root/HowTo.html>
- \$ROOTSYS/tutorials中的各个例子
- <http://hep.tsinghua.edu.cn/training/index.html>（一个三段的视频教程）