ROOT 学习笔记

> ROOT 是粒子物理与核物理数据分析的好工具！！！
> ROOT的学习不是一朝一夕的事情,需要反复反复再反复使用，才可能较好地掌握它.
> 这里是我学习使用ROOT的总结、感悟. 本文档的出发点是给初学者提供一种学习ROOT的思路如果C++基础好,学习ROOT会很快上手！这里简单介绍ROOT里面几个
> 最常用到的类,以及这些类的基本操作方法对于一些重要的类,仔细研读源程序会
> 有很大收获！

> ROOT学习资料
> 1. ROOT_for_beginners // 个人觉得这是最适合新手的学习资料,一共5篇
> 2. 杨振伟老师ROOT课程讲义 // 适合新手入门
> 3. ROOT-User-Guide
> 4. $ROOTSYS/tutorials // tutorials源代码在ROOT安装目录tutorials下，是非常好的学习资料！
> 5. 新版本Reference-Guide
> 6. *旧版本Reference-Guide

> ROOT学习方法参考!!!
>
> 1. 入门阶段：建议阅读顺序，ROOT_for_beginners，杨振伟老师ROOT课程讲义, 完成里面的练习
> 2. 提高阶段: ROOT-User-Guide 与 tutorials 结合使用 (User-Guide不适合从头到尾阅读!!!)
> 3. 熟练阶段： 在root环境下善用Tag键不全，必要时查阅Reference-Guide

作者：小关

目录

ROOT 基础篇

ROOT-Framework简介

- $ROOTSYS/bin : 二进制文件:
- $ROOTSYS/lib : ROOT库文件 (写makefile时需要用到！！！)
- $ROOTSYS/tutorials: ROOT例子源代码
- $ROOTSYS/Test : 包含整个ROOT-Framework的全部实例,值得进一步探索！！！
- $ROOTSYS/include: 包含所有的头文件

ROOT 终端常用命令(更多内容参见cling)

```
root -h  //help作用，查看root后面参数如何使用
root -l  //关root的欢迎界面
root -b  //关闭图形界面，及不显示Canvas
root myMacro.C > myMacro.log  // 将 myMacro.C 的结果输出到 myMacro.log中

root[] .?             // 查看root环境下所有的用法
root[].L myFile.C     // Load myFile.C
root[].x myFile.C     // Load and execute myFile.C
//更多用法参照 cling 的介绍
```

ROOT的代码规范

代码约定

命名规则	代码规范
类名以 "T" 开头	TLine, TTree, ...
非类类型以"_t"结尾	Int_t, Double_t, Bool_t, ...
类的数据成员以"f"开头	fTTree, ...
成员函数以大写字母开头	Loop(), ...
常量以"k"开头	kRed, ...
全局变量以"g"开头	gROOT, gStyle, ...
静态数据成员以 "fg" 开头	fgTokenClient, ...
枚举型以 "E" 开头	EColorLevel, ...
局域变量与参数开头小写	nbytes, ...
Getters and Setters 分别以 "Get" "Set" 开头	SetLast(), GetFirst(), ...

数据类型规范

> 为避免新老机器对同一种数据类型可能有不同的长度, ROOT使用下面的 pre-defined 类型

```
* Char_t         //Signed Character 1 byte
* UChar_t        //Unsigned Character 1 byte
* Short_t        //Signed Short integer 2 bytes
* UShort_t       //Unsigned Short integer 2 bytes
* Int_t          //Signed integer 4 bytes
* UInt_t         //Unsigned integer 4 bytes
* Long64_t       //Portable signed long integer 8 bytes
* ULong64_t      //Portable unsigned long integer 8 bytes
* Float_t        //Float 4 bytes
* Double_t       //Float 8 bytes
* Double32_t     //Double 8 bytes in memory, written as a Float 4 bytes
* Bool_t         //Boolean (0=false, 1=true)
```

全局变量

gROOT

> By using gROOT pointer, you can get the access to every object created in a ROOT program

```
root[] gROOT->ProcessLine(".x myHist.C");
root[] gROOT->GetListOfFunctions();
root[] gROOT->GetListOfCanvases()->FindObject("c1");
...
```

gPad

> gPad is always pointing to the active pad

```
{
  gPad->SetFillColor(38);
  gPad->Modified(); // Tell the canvas that an object it is displaying has changed
  gPad->Update();  // Force the canvas to refresh
  ...
}
```

gStyle

```
root[] gStyle->SetFillStyle();
root[] gStyle->SetPalette(1);     // To plot with nice colors
root[] gStyle->SetOptFit(kTRUE);  // 显示拟合参数
root[] gStyle->SetOptStat(1);     // 显示详细的拟合参数
...
```

gRandom

> A pointor to the current random number generator. Points to 'TRandom3' by default

```
root[] gRandom->Print(); // 查看当前的 random number generator
root[] delete gRandom;   // 删除当前的 random number generator
root[] gRandom = new TRandom2(0); // seed = 0, 新的random number generator
...
```

gSystem

```
root[] gSystem->Getenv("USER")  // returns the value of the system enviroment variable '
```

其他全局变量

> 在 root 终端键入g, 按 Tab 补全可查看所有的全局变量!

Environment Setup

rootlogon.C

> This script without a function declaration is executed automatically when ROOT is launched from the same directory as the file

```
{
   gStyle->SetPalette(1);                                    // 使画图颜色更加好看
   cout << "Salut " << gSystem->Getenv("USER") << "!" << endl;
   gSystem->Exec("date");                                    // 显示系统时间日期
}
```

rootlogoff.C

> rootlogoff.C is a script loaded at shutdown

rootalias.C

> rootalias.C file is loaded but not executed at start-up, it contians small functions like:

```
ls(path)
edit(filename)
dir(path)
pwd()
cd(path)
```

对象

Inspecting Objects

```
root[] TFile f("staff.root");
root[] f.Inspect()
root[] f.Print()
```

Object Ownership

```
2.1 By Current Directory (gDirectory)
    所有权归当前目录的有: histograms, tree, event list(TEventList)

    TH1F *h = (TH1F*)gDirectory->GetList()->FindObject("myHist");


2.2 By the Master TROOT Object (gROOT)
    所有权归gROOT的有: 一些列 "collections of objects",比如 fCanvases,  fColors,...

    TCanvas *cc = (TCanvas*)gROOT->GetListOfCanvases()->FindObject("c1");


2.3 By Other Objcets
    When an object creates another, the creating object is the owner of the created one

    myHisto->Fit("gaus");


2.4 By the user
```

ROOT中的C++

C++ 解释器 -- Cling

> - Cling 是 ROOT 使用的 C++ 解释器. Cling 可以简化我们在root环境下的C++语法!

> - Cling 是解释器, 不是编译器！ 它给我们在 root 环境下使用 C++ 带来便利! 比如：root 可以直接执行 ROOT 脚本(也叫"Macro")而不需要编译， 这样的 macro 甚至不需要包含必要的头文件， 但且要求文件名与函数同名！
> - ROOT Macro 一般不能通过C++编译!!! 所以在写需要编译的复杂程序是不能使用 cling 带来的这些便利！ 切记！
> - 链接到cling

1. 解释器命令以"."开头, 在root终端可产看所有的命令

```
root[] .?  // 查看所有的命令
```

2. 命令行模式使用多行代码：以 "{" 开头,以 "}" 结尾

```
root[] {
root[] ? for(int i=0; i<5; i++){
root[] ?    cout<< i << endl;
root[] ?}
```

3. ROOT脚本的执行

> ROOT script files 通常也叫作 "Macros". 可以在一个脚本中执行另一个脚本.

```
// calls a script to build the root file if it does not exist
void cernstaff()
{
  if(gSystem->AccessPathName("cernstaff.root")) // 如果"cernstaff.root"不存在，则返回 true
  {
    gROOT->ProcessLine(".x cernbuid.C");
  }
}
```

ACLiC: Compiling Scripts Into Libraries

1. 使用方法

```
root[] .L MyScript.C+  // build and load a shared library containing your script

gROOT->ProcessLine(".L MyScript.C+");
```

2. 设置头文件路径

```
root[] .include  // get the include path
root[] .include $HOME/mypackage/inlcude  // append to the include path

gSystem->AddIncludePath(" -I$HOME/mypackage/inlcude");// 在脚本中添加
gSystem->SetIncludePath(" -I$HOME/mypackage/inlcude"); // overwrite the existing include
gSystem->AddLinkedLibs("-L/my/path -lanylib"); // Add library
gSystem->Load("mydir/mylib"); // Load library
```

GUI 图形用户界面

TCanvas && TPad

> TCanvas 与 TPad 的关系
>
> - TCanvas 是 TPad 的子类. 一个 canvas 本身是一个大 pad, 这个大的 pad 可以分为多个小 pad
> - 任何时候，只能有一个 pad 处于 active 状态, 画图也将画在 active 的 pad 上

TCanvas

```
TCancas *c1 = new TCanvas("name","title",width, height); // 创建新的canvas
c1->SaveAS();  // 保存
c1->Print();   // 保存
```

TPad

```
{
  gPad->SetLogy();            // 设置Log坐标
  gPad->Modified();           //
  gPad->Update();
  gPad->SetLogy(1);           // 设置对数坐标
  gPad->SetTicky(1);          // 给坐标轴设置网格
  gPad->SetLeftMargin(0.15);  //设置 pad 的偏置
  gPad->Modified(); // Tell the canvas that an object it is displaying has changed
  gPad->Update();  // Force the canvas to refresh
}
```

Folders and Tasks

Folders

> To reduce class dependencies and improve modularity

1. 创建文件夹

```
{
   // Add the top folder of my hierary to //root
   TFolder *aliroot=gROOT->GetRootFolder()->AddFolder("aliroot",
                                 "aliroot top level folders");
   // Add the hierarchy to the list of browsables
   gROOT->GetListOfBrowsables()->Add(aliroot,"aliroot");

   // Create and add the constants folder
   TFolder *constants=aliroot->AddFolder("Constants",
                                       "Detector constants");
}
```

2. 在文件夹添加内容 (Producer)

```
TObjArray *array;
run_mc->Add(array);
```

3. 从文件夹读取内容 (Consumer)

```
conf=(TFolder*)gROOT->FindObjectAny("/aliroot/Run/Configuration");
// or ...
conf=(TFolder*)gROOT->FindObjectAny("Configuration");
```

Tasks

Input/Output

ROOT 功能篇

Histograms 直方图

TH1

- 从已有root文件中读取histogram

```
TFile * in = new TFile("文件路径");
TH1F  * h1 = (TH1F*)in->Get("ObjectName");
TF1F  * h1 = (TH1F*)gROOT->FindObject("ObjectName"); //在ROOT环境下使用
```

- 创建并保存root文件

```
h1->GetNbinsX();          // get the number of bins in X axis
h1->GetBinCenter(i);      // get the center of bin NO.i
h1->GetBinContent(i);     // get the Y value of bin NO.i
h1->GetEntries();         // get the number of entry
```

- 直方图有用的用法

```
TH1F *hist = (TH1F*)h1->Clone();  // 克隆一个直方图
h->Scale(1./h->Integral());       // 归一化
```

- 直方图的画图技巧

```
hs->GetXaxis()->SetNdivisions(-505);   // 设置坐标值分度值
h->SetStats(0);                        // 关闭直方图右上方显示的box
h->SetOptStat(0);                      //
h->GetListOfFunctions()->Add(func); h->Draw(); // Draw the histo with the fit function
```

TH2

THStack

- 同时画出多个直方图：THStack

```
THStack *hs = new THStack("hs","title");
hs->Add(h1);
hs->Add(h2);
```

Graphs 画图

TGraph2DErrors

> I use TGraph2DErrors() to draw data(error value equal to 0), i try to fit with TF2 function, error happens: "fill data empty" // Reason: Reason: TF2 fit ignore data without an error

TLatex

TLegend

- How to add a legend to a figure

```
* 新建一个TLegend: TLegend * legend = new TLegend();
* Fit function linked to the hist: TF1 * fun = hist->GetFunction();
* 创建legend内容 :      sprintf(message,"#chi^{2}=%.2f", fun->GetChisquare())
* AddEntry:            legend->AddEntry(fun,message);
* Drawing the TLegend: legend->Draw();
```

TLine

Fitting 拟合

> Link-to-Root-User's-Guide

```
 gStyle->SetOptFit(kTRUE);           // 显示拟合参数
 hist->Fit("gaus","V","E1",-1,1.5);
// Fit("function name","fit options","drawing options",fit limits)
```

Use_predefined_funtion 使用自带函数

```
* Root 中自带的四类拟合函数:  "gaus","expo","polN","landau"
* 获取拟合参数
  Get the function: TF1 * gfit = (TF1*)h->GetFunction("gaus");
  Get the parameters:
  gfit->GetParameter(0);
  gfit->GetParameter(1);
  gfit->GetParError(0);
  ......
  double par[3];
  gfit->GetParameter(par);
```

Use user-defined function 使用自定义函数

> 自定义函数必须初始化才能使用

```
* Define the function
* Include it in a TF1
* Set parameters  : mw->SetParNames(); mw->SetParameter(1); mv->SetParameters(par);
* Make the fit
* Sensitive to the initial values: mw->SetParLimits(0,lowlimit, highlimit);
* Get fit results : mw->GetChisquare();mw->GetNDF(); // Number of Degrees of Freedom
```

Use mixing functions 使用混合函数

```
* Pre-defined functions : TF1 *fc=new TF1("f5","pol3(0)+[4]*sin(gaus(5)+[8])",0,10)
* User-defined functions:

    Double_t DeuxMaxwell(Double_t *x, Double_t *par)
    {
      /// Sum of 2 Maxwellian functions
      return Maxwell(x,&par[0])+Maxwell(x,&par[3]);
    }
```

注意两点：
a.使用自定义函数拟合时，拟合结果对参数初始化很敏感
b.一般需要给参数设定边界

Fittting options 拟合选项

```
* "Q"   Quite model,终端不输出拟合结果
* "V"   Verbose model, 详细的输出     <font color=#DC143C>//(默认的模式介于两者之间) </font>
* "R"   使用函数定义时给定的区间进行拟合 (用于多区间拟合)
* "+"   在不删除前一个函数的情况下，将当前的拟合函数添加到list里面 <font color=#DC143C>     //默
* "N"   不存储拟合函数，也不画图显示    <font color=#DC143C  > //(默认情况是既保存又画图) </font
* "0"   不画出拟合结果
* "LL"  An Improved Log Likelihood fit for low statistics   <font color=#DC143C> //(当Bi
```

Set Bounds for Parameters 拟合参数设置

```
func->SetParameter();    // 单独给某一个参数赋初值
func->SetParameters();   // 同时给所有的参数赋初值
func->SetParLimits();    // 给某一个参数设定边界
func->FixParameter();    // 固定某个参数
```

Get the associated function

```
* TF1 *myfunc = h->GetFunction("myfunc");  // 从直方图的拟合函数中提取
* Fit Statistics: gStyle->SetOptFit(mode)  mode = pcev (defaul = 0111)
      p=1   打印 probability
      c=1   打印 Chi2/NDF
      e=1   打印 errors (if e=1, v must be 1)
      v=1   打印参数 name/values
```

ROOT::Fit::Fitter ROOT6拟合新方法

应用举例

> - ROOT::Fit is a new ROOT Class in ROOT6
> - 相比于TH1::Fit， ROOT::Fit 能对Fit进行更多精细的操作和控制!
> - ROOT::Fit::BinData used for least chi-square fits of histograms or TGraphs
>
>   ROOT::Fit::UnBInData used for fitting vectors of data points (e.g. from a TTree)

```
{
// 1. Create the input fit data object
  TH1 * h1 = (TH1*)filein->Get("histName");
  ROOT::Fit::DataOptions opt;
  opt.fIntegral = true; // Use the integral of bin content instead of bin center(default
  ROOT::Fit::DataRange range(10, 50);
  // ROOT::Fit::DataRange range;
  // range.setRange(10, 50);
  ROOT::Fit::BinData data(opt,range);
  ROOT::Fit::FillData(data, h1);


// 2. Create the input model functon
   TF1 * f1 = new TF1("f1","guas");
   ROOT::Math::WrappedMultiTF1 fitfunc(*f1,f1->GetNdim());


// 3. Congidure the fit
   Double_t par[3] = {100, 30, 10};
   ROOT::Fit::Fitter fitter;
   fitter.setFunction(fitfunc, false);
   fitter.Config().SetParamsSettings(3, par);
   fitter.Config().ParSettings(4).Fix();
   fitter.Config().ParSettings().SetLimits(-10, -1.E-4);
   fitter.Config().ParSettings(3).SetLimits(0,10000);
   fitter.Config().ParSettings(3).SetStepSize(5);


// 4. Chose the minimizer
   fitter.Config().SetMinimizer("Minuit","Migrad");
   // To print the default minimizer


// 5. Perform the data fitting
   fitter.FitFCN(3, fitfunc, 0, data.Size(), true);

// 6. Examine the result
   ROOT::Fit::FitResult result = fitter.Result();
   result.Print(std::cout);


// 7. Draw
   f1->SetFitResult(result, par);
   f1->SetRange(range().first, range().second);
   h1->GetListOfFunctions()->Add(f1);
   h1->Draw();
}
```

FUMILI Minimization Package 最小化算法

> - To minimize Chi-square functio //(ROOT中默认的拟合方式是最小Chi2)
> - To search maximum of likelihood function

MINUIT

MINUIT2

利用神经网络进行数据拟合

Trees 树

ROOT 提高篇

Writing-GUI 手写GUI

Geometry Package

Python Interface

Networking

Threads 线程

Parallel-Processing 并行计算

ROOT 运算篇

Math-Libraries 数学库

Matrix 矩阵

Physics-Vectors 矢量运算

ROOT 其他篇

TCutG

> Int_t TCutG::IsInside(Double_t x, Double_t y) const

1. 判断一个点是否在给定Cut范围内

```
if(mycut->IsInside(x,y)==1) // (x,y) is inside the cut region
if(mycut->IsInside(x,y)==0) // (x,y) is outside the cut region
```

2. 读取已有的Cut与作新的Cut

```
TCutG cut = (TCutG*)gPad->GetPrimitive("CUTG")       // get a cut
TCutG * mycut = (TCutG*)gPad->WaitPrimitive("CUTG"); // draw a new cut
```

TList

```
TList * list = gPad->GetLIstOfPrimitives();   // List of objects in the current canvas
```