第二届(2021)高能物理计算暑期学校上机练习

一，作业提交及使用

1, 作业准备

- 登录集群

```
$ ssh username@schlogin.ihep.ac.cn
```

注意，username需替换为你的用户名

- 查看所在目录：

```
$ pwd
$ ls
```

- 创建作业脚本文件

```
$ touch job.sh
```

- 打开作业脚本

```
vim job.sh
```

- 编辑脚本内容

1. 复制下列内容

```
#!/bin/bash

/bin/sleep 10
echo "We're doing a simple operation:"
result=$(expr 1 + 1)
echo " 1+1=$result"
/bin/sleep 10
echo "Job Done!"
```

2. 按快捷键 i 进入编辑模式
3. 快捷键shift+insert粘贴内容至脚本文件中（或鼠标右键粘贴）
4. esc键退出编辑模式

- 保存并退出脚本编辑

```
:wq
```

- 设置脚本可执行权限

```
$ chmod +x job.sh
```

2, 作业提交

```
$ hep_sub job.sh
```

如果成功，显示内容如下：

```
1 job(s) submitted to cluster 13
```

其中，13代表作业id，作业id为作业最重要的身份信息，可利用作业id进行作业查询和删除等操作。

3, 作业查询

```
# 按用户查询
$ hep_q -u test001

# 按作业ID查询
$ hep_q -i 13

# 如果作业状态为'H', 表明作业被挂起，可使用如下命令
$ hep_q -i 13 -hold
$ hep_q -u test001 -hold
```

4, 作业删除

```
# 删除当前用户所有作业
$ hep_rm -a

# 按作业ID删除
$ hep_rm 13
```

5, 查看作业结果

- 作业结果文件
  如果查询作业时，没有查询到，表明作业已退出队列。查看作业结果

```
$ ls
```

找到 .out 和 .err 文件：

```
- .out 文件保存标准输出内容
- .err 文件保存标准错误内容
```

- 验证结果：

查看结果文件：

```
$ cat job.sh.out*
```

输出如下，说明作业正常运行结束

```
We're doing a simple operation:
 1+1=2
Job Done!
```

二，容器使用

1, 容器技术介绍

容器是一种轻量级、可移植、自包含的软件打包技术，使应用程序可以在几乎任何地方以相同的方式运行。无需任何修改就能够在生产系统的虚拟机、物理服务器或公有云主机上运行。

1.1 容器的特点

跨环境、可移植、资源和应用隔离性、安全性

1.2 容器与虚拟机区别

•容器直接运行在内核上

•容器优势启动快、高性能和低延迟

•虚拟机可以虚拟硬件不依赖内核、隔离更加彻底

1.3 容器引擎

Singularity是目前高通量（HTC）和高性能（HPC）计算平台上被大量应用的轻量虚拟化容器技术，能够提供操作系统级的虚拟化。

•Singularity 更轻、适合HPC，MPI(OpenMPI, MPICH, IntelMPI),GPU,infiniband

•Singularity 的用户权限容器内外都一致, /dev, /sys and /proc automount，安全性高

2, hep_container

Hep_container是基于singularity容器管理命令开发的适用于高能所计算集群的容器客户端工具，满足用户使用多种操作系统版本及环境的需求。 **说明**：本文涉及的命令均需要登陆节点上运行，所用命令在以下目录，建议将该目录加入用户个人环境变量 PATH 中。

```
/cvmfs/container.ihep.ac.cn/bin/
export PATH=$PATH:/cvmfs/container.ihep.ac.cn/bin/
```

Hep_container的特点

•hep_container容器统一运行入口程序

•用户对容器操作统一

•容器的参数和环境变量配置和镜像路径对用户透明

•容器更新不会影响到当前用户的使用

•实现同一容器在不同站点上运行不同的作业配置策略，高能所、中科大、北京大学、山东大学、兰州大学等站点多种作业调度配置，支持一平台多中心

•容器内用户只有自己实验组的数据盘访问权限，安全性高

2.1 命令说明

Hep_container的容器命令主要有以下操作images、shell、exec等。可以在命令行中通过help参数查看各个命令的使用说明和样例

```
$ hep_container  help
Usage : ./hep_container  <command> [command options...]
```

```
CONTAINER USAGE COMMANDS:
    shell           Run a Bourne shell within container image
    exec            Execute a command within container image
    images          List Support container images
    groups          List Support groups
    -g groupname    With a specific group name
EXAMPLES:
    ./hep_container images
    ./hep_container groups

    ./hep_container shell SL5
    ./hep_container shell SL5 -g physics

    ./hep_container exec  SL5 cat /etc/redhat-release
    ./hep_container exec  SL5 python ./yourprograme.py
    ./hep_container exec  SL5 -g physics cat /etc/redhat-release
```

2.2 查看支持镜像

命令格式：hep_container images 该指令可以查看当前提供的操作系统容器镜像。

```
$ hep_container images
Hep_container support images:
        SL5 : Scientific Linux 5
        SL6 : Scientific Linux 6
        SL7 : Scientific Linux 7
    CentOS7 : CentOS     Linux 7.8
    HepcMyImage : Custom Image file name
```

2.3 查看支持用户组/实验组

命令格式：hep_container groups 该指令可以查看容器命令当前支持提供的用户组或者实验组。通过 -g 参数指定用户组或实验组，容器内会挂载对应用户目录和实验目录。不指定-g参数默认采用主组作为用户组或实验组。

```
$ hep_container groups
Hep_container support groups:

u07|atlas|atlasrun|comet|offline|physics|higgs|ams|cms|dyw|hxmt|polars|juno|argo|lhaaso|sch
```

2.4 进入容器环境

命令格式：hep_container shell [container image] 该指令可以在容器内启动一个shell，因此可以在容器外部与容器内部进行交互操作。运行exit则可以退出该shell。 下例为运行启动一个SL5操作系统镜像后，用户当前为SL5的系统环境.

```
$ hep_container shell SL5
Singularity: Invoking an interactive shell within container...
Singularity> cat /etc/redhat-release
Scientific Linux SL release 5.5 (Boron)
Singularity> exit
exit
```

2.5 容器内执行命令

命令格式：hep_container exec [container image] [command] 该指令可以在外部主机上将指定的command运行在指定的容器内。 下例为在lxslc7上以SL5的环境运行SL5命令，并得到结果。

```
$ hep_container exec SL5 cat /etc/redhat-release
Scientific Linux SL release 5.5 (Boron)
```

3, 镜像制作

3.1 通过定义文件制作镜像

```
[user1@dockertest02]# cat myimage.def
BootStrap:yum
OSVersion: 7.8

   MirrorURL: http://mirror.ihep.ac.cn/centos/7/os/x86_64/

   UpdateURL: http://mirror.ihep.ac.cn/centos/7/os/x86_64/

   Include: yum

%setup

%files

   #~/home/yourfile  ~/usr/local/yourfile

%runscript

  echo "Running the container..."

%post

    echo "Installing base group"
    yum -y groupinstall "Minimal Install"
    echo "Installing basic packages"
    yum -y install vim-enhanced wget ntp gcc gcc-c++ glibc make
    echo "Installing requied packages"
    yum -y install python36

```

```
sudo singularity build myimage.sif myimage.def
```

3.2 暑期学校使用个人镜像使用流程

```
export MYIMAGE=/home/sch/test001/mymkimage.sif
hep_container shell MYIMAGE
```

3.3 高能物理集群个人镜像流程

由于安全原因，个人镜像不能随意在集群中发布，个人镜像需要进行严格审核后（需求审核、用户审核，安全审核）方可使用

3.4 其他镜像制作方法

参考 https://sylabs.io/guides/3.5/user-guide/build_a_container.html

三，Git操作练习

1, git本地配置

- 配置用户、邮箱及编辑器
  如果不对 git 做全局设置的话，可以去掉 `--global` 选项。

```
# 请修改your name为自己的名字
$ git config --global user.name "your name"
# 请修改your_email_address为自己的邮箱地址
$ git config --global user.email your_email_address
$ git config --global core.editor vim
```

- 查看配置文件及选项

```
$ git config --list
```

- 设置 alias (可选)

```
[user]
        name = "Your-name"
        email = "Your-email"

[alias]
        a = add .
        v = !gitk
        br = branch
        ci = commit -m
        cm = commit -m
        co = checkout
        df = diff
        dump = cat-file -p
        hs = log --pretty=format:\"%h %ad | %s%d [%an]\" --graph --
date=short
        last = log -1 HEAD
        pl = pull
        ps = push
        st = status
        type = cat-file -t
        sum = shortlog -sn

[push]
        default = matching
```

- ssh 配置(可选)

```
#### ~/.ssh/config
Host code
        Hostname code.ihep.ac.cn
        User git
        Identityfile ~/.ssh/id_rsa
```

2, 准备远程及本地代码仓库

```
$ mkdir -p ~/source
$ rsync -avru /cvmfs/slurm.ihep.ac.cn/2021_summer_school_git/marigold
~/source/
$ cd ~/source/marigold
$ git config receive.denyCurrentBranch ignore
$ cd ~
$ git clone ~/source/marigold
```

3, 在本地仓库目录下操作

```
$ cd ~/marigold
# 编辑修改该目录下的文件README
$ echo "1. Modified to check file status after edit." >> README.md
# 编辑后查看文件状态
$ git status
```

4, 暂存操作

```
# 将修改的文件README.md暂存
$ git add README.md
# 查看文件状态
$ git status
```

5, 提交操作

```
# 提交暂存区的文件
$ git commit -m 'Add one line on README.md.'
# 查看文件状态
$ git status -s
```

6, 远程仓库操作

```
# 查看远程仓库
$ git remote -v
# 发布到远程仓库(一般 push 分支)
$ git push origin master
```

7, 标签操作

```
# 给自己的稳定版本增加版本标签吧
$ git tag -a v1.0 -m 'Basic version'
# 列出所有标签
$ git list
# 查看标签的详细信息
$ git show v1.0
# 共享标签
$ git push origin v1.0
```

8, 分支操作

```
# 查看分支
$ git branch
# 创建分支
$ git branch chkbr
# 切换到新建的branch
$ git checkout chkbr
# 查看分支
$ git branch
# 修改文件README.md
echo "2. Add a new branch named chkbr for branch operation practice." >>
README.md
# 将修改后的文件放置暂存区
$ git add README.md
# 提交修改后的文件
$ git commit -m 'Modify README.md - practise branch operations.'
# push 分支到远程仓库
$ git push origin chkbr
# 切换回master分支
$ git checkout master
# 合并分支chkbr
$ git merge chkbr
# 查看branch
$ git branch
# 查看操作日志
$ git log --oneline --decorate --graph -all
# 删除分支
$ git branch -d chkbr
# 查看分支
$ git branch
```

四，ROOT 分析作业

source 环境

```
> cd ~
> source /cvmfs/sft.cern.ch/lcg/releases/LCG_84/ROOT/6.06.02/x86_64-slc6-
gcc48-opt/bin/thisroot.sh
```

拷贝和查看练习示例脚本

```
> cd ~
> cp -r /cvmfs/slurm.ihep.ac.cn/2021_summer_school_root/ .
> cd 2021_summer_school_root/
> ls
-rw-r--r-- 1 user group  1792 Aug 16 19:45 exercise0.C
-rw-r--r-- 1 user group   969 Aug 16 19:49 exercise1.C
-rw-r--r-- 1 user group  2848 Aug 16 19:40 exercise2.C
```

1，生成数据样本

1.1 定义分布函数(信号+本底)

Lorentzian Peak function:

$$Loren(x) = \frac{0.5 \cdot a0 \cdot a1}{\pi} \cdot \frac{1}{Max[(x-a2)^2 + \frac{a1^2}{4}, 10^{-10}]} + (b0 + b1 \cdot x + b2 \cdot x^2)$$

```
   // Lorentzian Peak function
   Double_t lorentzianPeak(Double_t *x, Double_t *par) {
      return (0.5*par[0]*par[1]/TMath::Pi()) / TMath::Max(1.e-10,
      (x[0]-par[2])*(x[0]-par[2])+ .25*par[1]*par[1]);
   }

// Quadratic background function
   Double_t background(Double_t *x, Double_t *par) {
      return par[0] + par[1]*x[0] + par[2]*x[0]*x[0];
   }

   // Sum of background and peak function
   Double_t Function(Double_t *x, Double_t *par) {
      return background(x,par) + lorentzianPeak(x,&par[3]);
   }
```

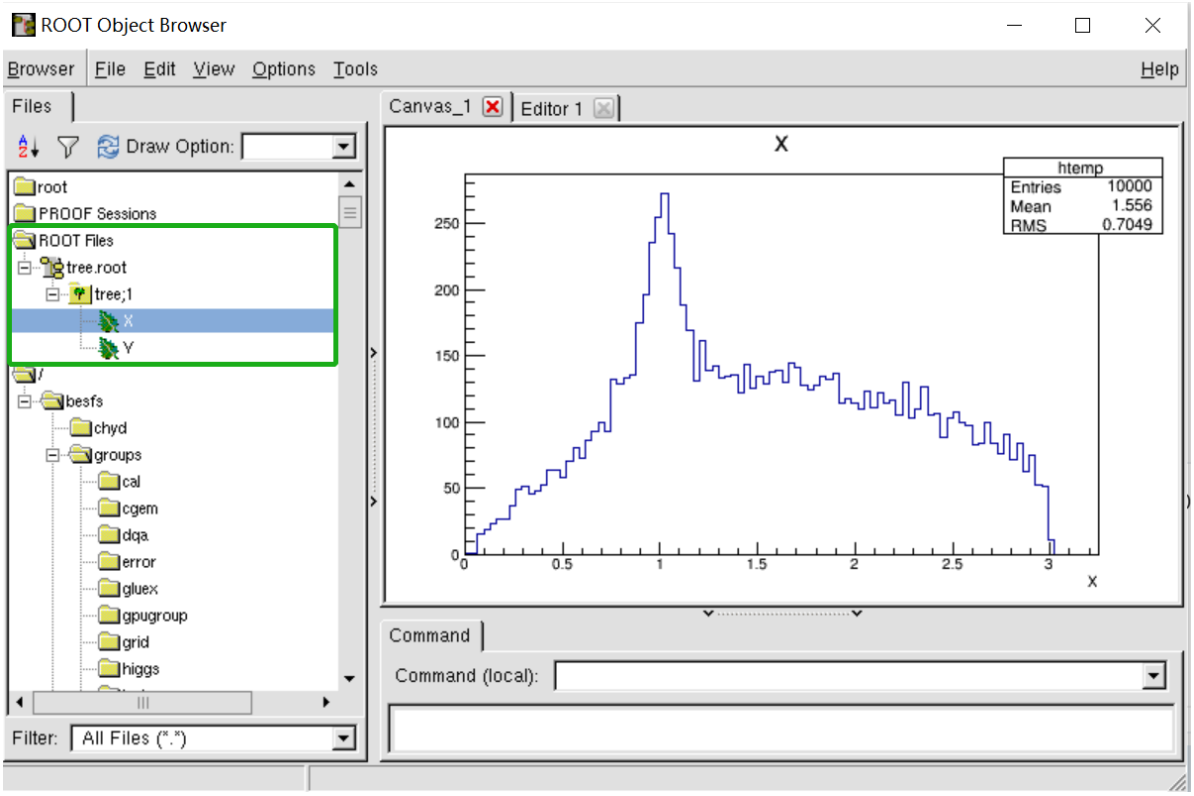
1.2 生成数据样本，填tree，写入root文件

```
   // create the file, the Tree and a few branches
   TFile f = TFile::Open("tree.root","recreate");
   TTree *tree = new TTree("tree","treelibrated tree");
   Float_t x, y;

   // create a branch with energy
   tree->Branch("X",&x);
   tree->Branch("Y",&y);

   // create a TF1 with the range from 0 to 3 and 6 parameters
   TF1 *Fcnx = new TF1("Fcnx", Function, 0,3,6);
   TF1 *Fcny = new TF1("Fcny", Function, 0,3,6);

   // Fix the Parameters of function
   Fcnx->SetParameters(-1, 45, -13.3, 13.8, 0.2, 1);
   Fcny->SetParameters(-1, 45, -13.3, 13.8, 0.1, 1.5);

```

```
    // fill some events with random numbers
    Int_t nevent=1000;
    for (Int_t iev=0;iev<nevent;iev++) {
        x = Fcnx->GetRandom();
        y = Fcny->GetRandom();
        tree->Fill();   // fill the tree with the current event
    }

    // save the Tree header. The file will be automatically closed
    // when going out of the function scope
    tree.Write();
```

操作练习

```
> root -l exercise0.C
root [0]
Processing exercise0.C...
root [1] .q
# 查看生成的 root 文件
> ls -l
-rw-r--r-- 1 user group  1792 Aug 16 19:45 exercise0.C
-rw-r--r-- 1 user group   969 Aug 16 19:49 exercise1.C
-rw-r--r-- 1 user group  2848 Aug 16 19:40 exercise2.C
-rw-r--r-- 1 user group 76796 Aug 16 20:03 tree.root

# 打开 root 文件，通过 TBrowser 查看内容
> root -l tree.root
root [0] TBrowser a
# 在弹出窗口左侧文件列表中 "ROOT Files" 目录下找到 tree.root 文件，
# 双击 root 文件查看tree "tree"，双击 tree 查看 Branch "X" 和 “Y”，
# 双击 Branch “X” 可以在右侧窗口中看到 “X” 的直方图。
```

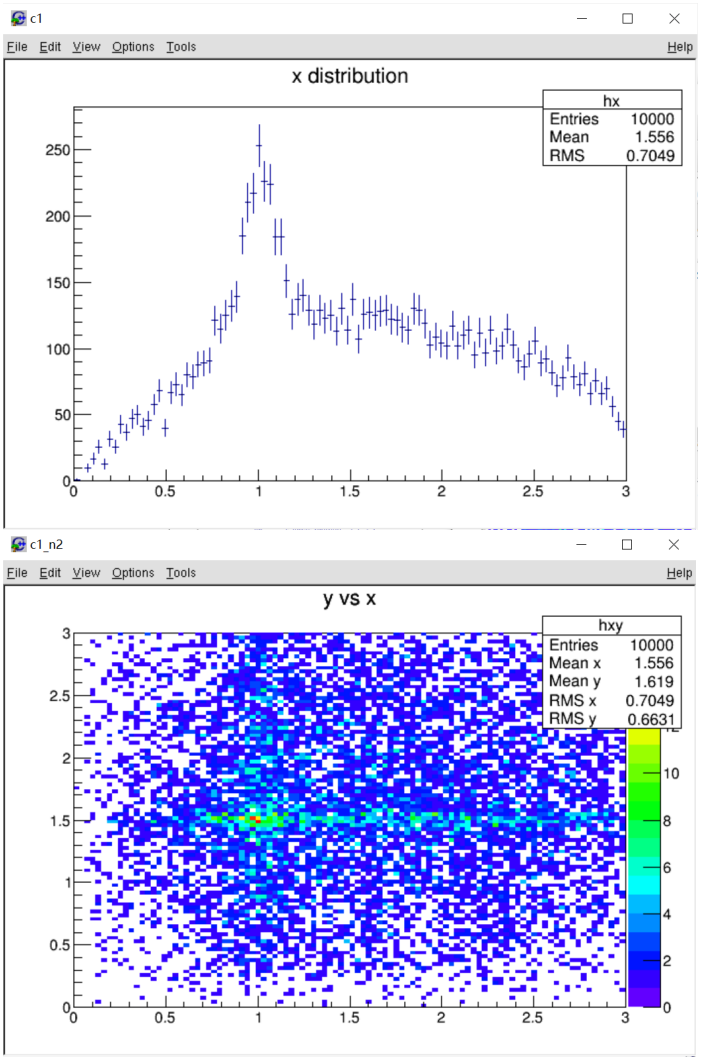


2. 读取root文件，填直方图，画出直方图

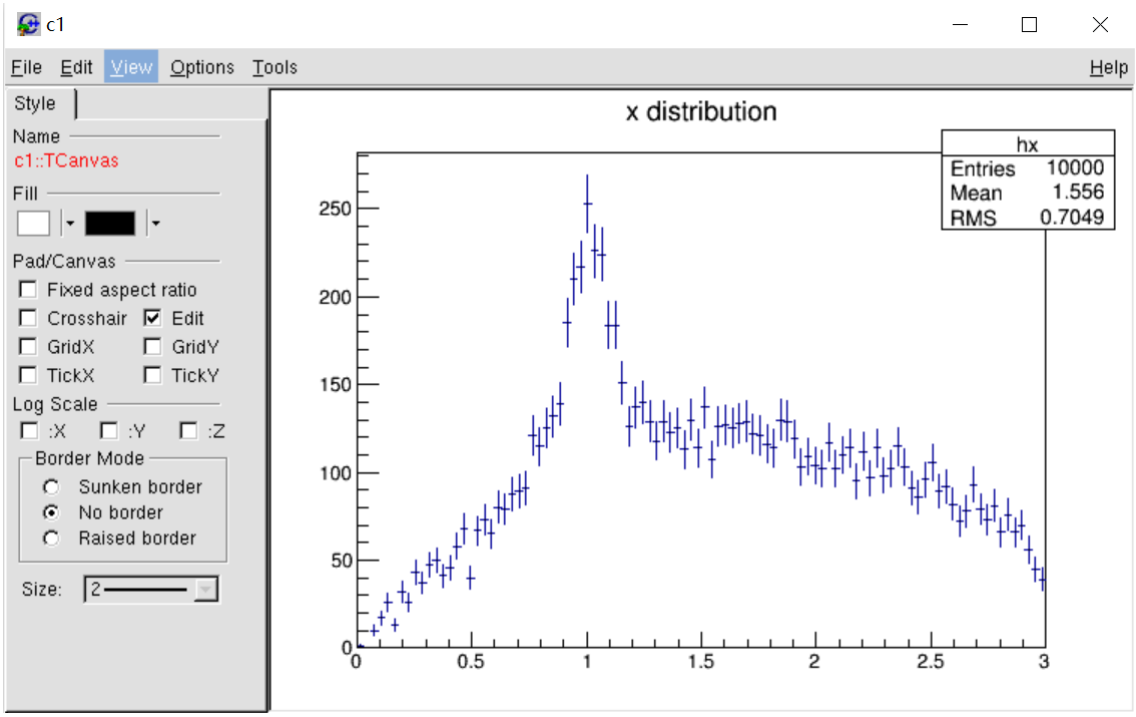
```
    TFile *f = TFile::Open("tree.root");
    TTree *t1 = (TTree*)f->Get("tree");

    Float_t x, y;
    t1->SetBranchAddress("X",&x);
    t1->SetBranchAddress("Y",&y);

    // create two histograms
    TH1F *hx  = new TH1F("hx","x distribution",100,-3,3);
    TH2F *hxy = new TH2F("hxy","y vs x",30,-3,3,30,-3,3);

     // read all entries and fill the histograms
     Long64_t nentries = t1->GetEntries();
     for (Long64_t i=0;i<nentries;i++) {
         t1->GetEntry(i);
         hx->Fill(x);
         hxy->Fill(x,y);
     }

     // draw the histograms
     TCanvas *c1 = new TCanvas();
     hx->Draw("E0");
     c1->Print("hx.png")  // eps, ps, jpg

     TCanvas *c2 = new TCanvas();
     hxy->Draw("colz"); // BOX, SCAT, ARR
     c1->Print("hxy.png")
```

操作练习

```
> root -l exercise1.C
root [0]
# 可以看到生成了一个一维直方图以及二维直方图
# 尝试改变画图选项，得到不同风格的直方图
```



单击菜单栏中 View 下拉菜单选中 Editor 选项可以调出编辑面版，可以手动编辑图片属性，如坐标轴标题，添加网格，直方图颜色等。尝试用鼠标单击图像不同位置，如直方图，坐标轴，标题处，观察编辑面板编辑选项的变化。尝试手动编辑图像属性。



3. 拟合直方图

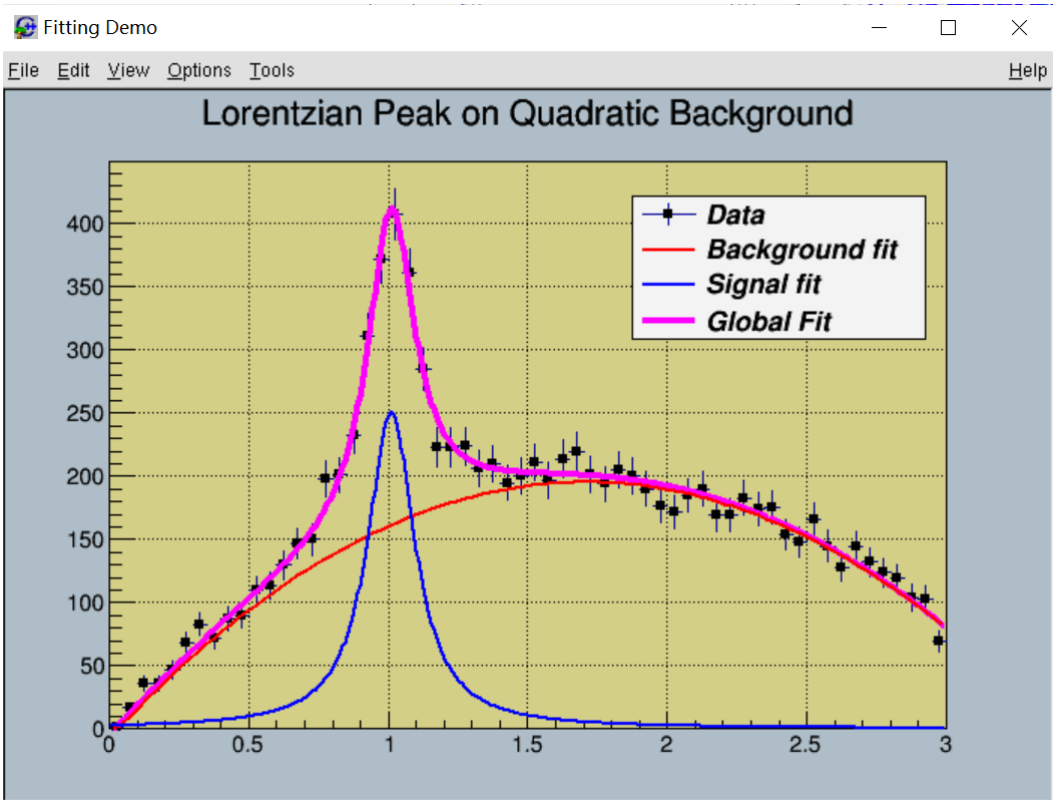
```
    // Read the Tree
    TFile *f = TFile::Open("tree.root");
    TTree *t1 = (TTree*)f->Get("tree");

    TCanvas *c1 = new TCanvas("c1","Fitting Demo",10,10,700,500);
    c1->SetFillColor(33);
    c1->SetFrameFillColor(41);
    c1->SetGrid();

    // Create one histogram
    TH1F *histo = new TH1F("histo", "Lorentzian Peak on Quadratic
Background",60,0,3);
    histo->SetMarkerStyle(21);
    histo->SetMarkerSize(0.8);
    histo->SetStats(0);

    // Fill the histograms
    t1->Project("histo", "X");

    // create a TF1 with the range from 0 to 3 and 6 parameters
    TF1 *fitFcn = new TF1("fitFcn",fitFunction,0,3,6);
    fitFcn->SetNpx(500);
    fitFcn->SetLineWidth(4);
    fitFcn->SetLineColor(kMagenta);

    // first try without starting values for the parameters
    // This defaults to 1 for each param.
    // this results in an ok fit for the polynomial function
    // however the non-linear part (lorenzian) does not
    // respond well.
    fitFcn->SetParameters(1,1,1,1,1,1);
    histo->Fit("fitFcn","0");

    // second try: set start values for some parameters
    fitFcn->SetParameter(4,0.2); // width
    fitFcn->SetParameter(5,1);   // peak

    histo->Fit("fitFcn","V+","ep");
    // improve the picture:
    TF1 *backFcn = new TF1("backFcn",background,0,3,3);
    backFcn->SetLineColor(kRed);
    TF1 *signalFcn = new TF1("signalFcn",lorentzianPeak,0,3,3);
    signalFcn->SetLineColor(kBlue);
    signalFcn->SetNpx(500);
    Double_t par[6];

    // writes the fit results into the par array
    fitFcn->GetParameters(par);

    backFcn->SetParameters(par);
    backFcn->Draw("same");

    signalFcn->SetParameters(&par[3]);
```

```
53      signalFcn->Draw("same");
54
55      // draw the legend
56      TLegend *legend=new TLegend(0.6,0.65,0.88,0.85);
57      legend->SetTextFont(72);
58      legend->SetTextSize(0.04);
59      legend->AddEntry(histo,"Data","lpe");
60      legend->AddEntry(backFcn,"Background fit","l");
61      legend->AddEntry(signalFcn,"Signal fit","l");
62      legend->AddEntry(fitFcn,"Global Fit","l");
63      legend->Draw();
```

操作练习

```
1  > root -l exercise1.C
```

得到拟合结果：



可以在输出 log 里面找到函数参数的拟合值：

```
1  NO.   NAME      VALUE            ERROR       NEGATIVE      POSITIVE
2     1  p0          -6.98862e+00   1.03331e+00
3     2  p1           2.36951e+02   5.46818e+00
4     3  p2          -6.93593e+01   2.12246e+00
5     4  p3           8.09132e+01   5.59809e+00
6     5  p4           2.05657e-01   1.77811e-02
7     6  p5           1.01006e+00   5.72858e-03
```

也可以通过以下方式拿到单个参数值及其误差：

```
# Get Associated Function
root[0] TF1 *fit = histo->GetFunction("fitFcn");
# value of the first parameter
root[1] Double_t p1 = fit->GetParameter(0);
# error of the first parameter
root[2] Double_t e1 = fit->GetParError(0);
# 同时我们还可以计算拟合的 chisquare 值
root[3] Double_t chi2 = fit->GetChisquare();
```

附加练习

尝试将本例中的朗道分布替换为高斯分布，完成 root 文件生成，root 文件读入，画直方图，拟合直方图。

$$Gauss(x) = a * e^{-\frac{(x-b)^2}{2*c^2}}$$

五，CMAKE编译

拷贝和查看练习示例脚本

```
> cd ~
> cp -r /cvmfs/slurm.ihep.ac.cn/2021_summer_school_cmake .
> cd 2021_summer_school_cmake/
> ls
-rw-r--r-- 1 user group  1792 Aug 16 19:45 cmake1
-rw-r--r-- 1 user group   969 Aug 16 19:49 cmake2
-rw-r--r-- 1 user group  2848 Aug 16 19:40 cmake3
```

1. 编译可执行文件

1.1 命令行编译

C++源代码

```
#include <iostream>

using namespace std;


int main(int argc,char** argv)
{
    cout << "Hello for cmake!" << endl;
    return 0;
}
```

操作练习

```
g++ -o helloworld helloworld.cpp
```

1.2 Make编译

Makefile

```
helloMake: helloworld.cpp
    g++ helloworld.cpp -o helloMake

clean:
    rm helloMake  -rf
```

操作练习

```
make
```

1.3 CMake编译

CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8.11)
project(HELLO)
add_definitions(-w)

add_executable(helloworld helloworld.cpp)
```

操作练习

```
mkdir build
cd build
cmake ..
make
```

2. 编译动态链接库

C++源代码

```
#include <iostream>
#include "sayhello.hpp"


void say_hello()
{
    std::cout << "Say Hello from DL builded by CMake" << std::endl;
}
```

CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8.11)
project(SayHelloLib)

include_directories("sayhello")
add_library(sayhello SHARED sayhello.cpp)


INSTALL(TARGETS   sayhello LIBRARY DESTINATION ${PROJECT_SOURCE_DIR}/../lib)
INSTALL(DIRECTORY sayhello DESTINATION ${PROJECT_SOURCE_DIR}/../inc)
```

操作练习

```
mkdir build
cd build
cmake ..
make
```

3. 链接动态链接库

C++源代码

```
#include <iostream>
#include "sayhello/sayhello.hpp"

using namespace std;

int main(int argc,char** argv)
{
    cout << "Hello for cmake!" << endl;
    say_hello();
    return 0;
}
```

环境变量设置

```
export PATH=/home/tianhl/workarea/CMakeWorkshop/bin:$PATH
export
LD_LIBRARY_PATH=/home/tianhl/workarea/CMakeWorkshop/lib:$LD_LIBRARY_PATH
```

CMakeLists.txt

```
cmake_minimum_required(VERSION 2.8.11)
project(CallHELLO)

link_directories(${PROJECT_SOURCE_DIR}/../lib)
include_directories(${PROJECT_SOURCE_DIR}/../inc)
add_executable(helloDL helloDL.cpp)

target_link_libraries(helloDL sayhello)

INSTALL(TARGETS helloDL DESTINATION ${PROJECT_SOURCE_DIR}/../bin)
```

操作练习

```
mkdir build
cd build
cmake ..
make
```