

# Пакет ROOT, часть 1

Значительную часть работы научного сотрудника составляет представление своих результатов в графическом виде. Поэтому мощный, специально приспособленный к особенностям физики высоких энергий, пользующийся преимуществами языка C++ графический пакет *ROOT*[2] является востребованным инструментом и теоретиков, и экспериментаторов, несмотря на сохраняющиеся в нем недоделки и ошибки кода.

## Настройка переменных окружения

Мы предполагаем, что *ROOT* уже установлен и Вы знаете его расположение. Например, на вычислительной *Linux*-ферме ОИЯИ (*lxpubXX.jinr.ru*) версии *ROOT* расположены в директории `/usr/local/root/pro/`, на ферме в ЦЕРНе (*lxplusXXX.ЦЕРН.ch*) *ROOT* в директории `/afs/ЦЕРН.ch/cms/external/lcg/external/root/`, на *Linux*-сервере филиала МГУ (*lx.msu.dubna.ru*) текущая версия находится в директории `/opt/sw/root/pro`

Сначала проверьте, может быть, ваша среда уже настроена для использования *ROOT*, просто попробуйте в командной строке ввести команду: `root`. Если ваша система не находит *ROOT*, значит, нужно установить некоторые переменные окружения. Как это делается, зависит от версии *shell*, которую Вы используете. Определить версию можно, набрав команду `finger` (ниже пример использования):

```
host>finger belotel
Login: belotel           Name: Ivan Belotelov
Directory: /home/belotel Shell: /bin/tcsh
host>
```

Для варианта **tcsh(csh)** отредактируйте ваш файл **.cshrc**, находящийся в вашей домашней директории (или создайте его, если он отсутствует). Добавьте туда следующие строки (приведён пример для *lx.msu.dubna.ru*):

```
setenv ROOTSYS /opt/sw/root/pro
setenv PATH ${ROOTSYS}/bin:${PATH}
setenv LD_LIBRARY_PATH ${ROOTSYS}/lib:$
{LD_LIBRARY_PATH}
```

Для **bash(sh)** отредактируйте (или создайте) файл **.bash\_profile**, добавив туда следующие строки:

```
export ROOTSYS=/opt/sw/root/pro
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:
$ROOTSYS/lib
export PATH=$PATH:$ROOTSYS/bin
```

При следующем входе в систему только что отредактированные начальные скрипты выполнятся и переменные среды установятся нужным образом. Чтобы начать работать прямо сейчас, выполните команду **source .cshrc** либо **source .bash\_profile**. Теперь попробуйте ещё раз запустить *ROOT*. Если всё в порядке, вы должны увидеть рисунок-заставку и строку приглашения. Теперь мы можем начать пользоваться пакетом *ROOT*.

## **Гистограммы**

Гистограммы — это основной графический объект, возможности работы с которым предоставляет *ROOT*. Гистограмма является графиком, предназначенным для отображения распределений количества физических событий по интересующим параметрам этих событий. Экспериментальные данные отображаются на гистограмме следующим образом. Допустим, было набрано 1000 событий и необходимо посмотреть, как распределились эти события по некоему одному параметру (например, по суммарной энергии зарегистрированных в единичном

события частиц). Тогда нужно создать гистограмму, в которой по оси  $X$  будет располагаться величина этой суммарной энергии, а по оси  $Y$  количество распределенных событий, такая гистограмма называется одномерной. При создании гистограммы необходимо определить, какова будет область определения этой энергии (например, от 0 до 5 ТэВ) и на какое количество частей эта область будет разделена. Такие части называются бинами гистограммы; если, например, определено, что бинов будет 50, это означает, что на гистограмме будет отображено 50 экспериментальных точек. Если определить, что бины должны быть равной ширины, то первый бин будет со значения 0 ТэВ и заканчиваться значением 0.1 ТэВ, второй бин будет 0.1-0.2 ТэВ и т.д. При необходимости можно создавать гистограммы и с разной шириной бинов. Итак, если из 1000 событий в диапазон первого бина попало, например, 9 штук, то первая отображаемая точка гистограммы будет иметь координаты (0.0, 9), если в диапазон второго бина попало 18 событий то координата второй точки (0.1, 18) и т.д.

Таким образом, гистограмма предназначена для работы со статистическими данными, и важным отличием её от других графиков является возможность заполнять её постепенно, по мере того, как новые статистические данные становятся доступны.

Одномерные гистограммы нужны прежде всего для количественной оценки фона и искомого эффекта, математического описания формы распределения эффектов, нахождения усредненных величин (в частности, нахождения массы частиц и резонансов). Поэтому скрупулезная работа с одномерными гистограммами необходима для получения конечных результатов экспериментов. В качестве примера этой важности, в частности, тщательной оценки уровня

фона можно привести произошедшее несколько лет назад открытие и «заккрытие» *пентаварка* [1] (Тема «*ROOT*», ресурс «*Выводы из истории поиска пентаварка*»)[3].

## Начало практической работы с гистограммами

В любом текстовом редакторе создаём файл с C++ кодом по имени, например, *example1.cxx*. Работая на сервере *sullen*, удобнее всего воспользоваться редактором *emacs*:

```
host>emacs example1.cxx &
```

Код для этого скрипта, который будет создавать и отображать 3 гистограммы, можно скопировать с сайта по адресу [1] (Тема «*ROOT*», ресурс «*Практика работы с гистограммами*»). Поясним те процедуры, которые относятся собственно к классам *ROOT*:

```
TH1F* hist1 = new TH1F("h1","hist. 1",100, 0., 10);
```

- конструктор гистограммы с именем *h1*, с отображаемым заголовком (title) «*hist. 1*», количество бинов 100, область определения 0-10.

```
hist1->SetLineColor(kRed);
```

 -настройка отображения  
линии гистограммы красным цветом.

```
TRandom* generator = new TRandom();
```

```
hist1->Fill(generator->Gaus(5, 0.5));
```

 - создание генератора псевдослучайного числа и заполнение единичным событием с распределением Гаусса.

```
hist3->Add(hist2);
```

 - прибавление к *hist3* содержимого гистограммы *hist2*.

```
TCanvas * c = new TCanvas("c", "c", 300, 10, 400, 700); c->Divide(1,3);
```

 - создание канвы, т.е. окна, в котором графически будут отображаться гистограммы с разрешением *400x700 точек*. Разделение на три подканвы для отображения трех гистограмм.

```
hist1->Draw("same");
```

 - отображение гистограммы поверх

активного на данный момент графического объекта (опция **same**).

**c->Print("c1.pdf");** - сохранение изображения в экспортном формате *pdf*. В зависимости от расширения имени файла, можно заказать сохранение в форматах *eps*, *gif*, *png* и т.д. Файл формата *pdf* затем можно посмотреть с помощью команды: **acroread имя\_файла &**

Запускаем интерпретатор *ROOT* в командной строке: **root example1.cxx** . Или можно запустить *ROOT*, а уже потом из его собственной командной строки запустить и скрипт: **root [0].x example1.cxx**

Если всё сделано правильно, то отобразится окно (канва) с нашими гистограммами, которые можно сравнить с изображением в выше упомянутом ресурсе сайта. Сейчас можно попрактиковаться в настройке атрибутов гистограмм, кликая курсором на объекты (тителы гистограмм, заголовки осей) и выбирая соответствующие опции. Отметим, что наличие поясняющих подписей на осях является необходимым признаком профессионализма! Канву с настроенными объектами можно сохранить в файле формата *ROOT* или скрипта *C++*, если выбрать команду *File* в верхней строке канвы и затем соответствующую опцию. Сохраненные файлы можно загрузить и в новой сессии *ROOT* (выйти при помощи команды **.q** , запустить снова **root**). Для загрузки *ROOT* файла (*c1.root*) нужно вызвать *ROOT* браузер командой **TBrowser**, а в нем найти активировать мышью сохраненный файл. *ROOT* файлы позволяют получить канву со всеми настройками. *C++* скрипт (*c1.C*) при своем запуске (команда **.x c1.C** в командной строке *ROOT*) отобразит канву не с предыдущими, а с текущими настройками сессии, но зато файл *C++* полезен тем, что в его коде можно посмотреть, как те настройки, что были сделаны вручную, можно

сделать C++ средствами, а затем использовать этот код в других скриптах для автоматизации настроек.

Чтобы сохранить не канву с гистограммами, а самостоятельные гистограммы, нужно открыть *ROOT* файл и записать туда гистограммы (или любые другие объекты):

```
TFile *rootFile = new Tfile("test.root",  
"recreate"); hist1->Write(); hist2->Write();  
rootFile->Close();
```

Чтобы уже в другом скрипте (или даже в новой сессии *ROOT*) просмотреть содержимое файла, а затем считать объекты из файла, можно применить следующие команды:

```
TFile *rFile = new TFile("test.root");  
rFile->ls(); TH1F* hst1 = rFile->Get("h1");
```

## ***Работа с 2D гистограммами***

Если одномерные гистограммы позволяют получить количественную оценку зависимости, то многомерные гистограммы применяются для качественного поиска корреляций. В качестве примера использования 2D-гистограмм как ключевого экспериментального результата можно привести поиск 2011 года динамических эффектов кварк-глюонной плазмы коллаборацией CMS[1] (*Тема «ROOT», ресурс «Исследование корреляций на CMS, 2011»*) [4]. Для того, чтобы нащупать корреляции, необходимо уметь настроить наиболее информативное отображение гистограммы, учитывая что двумерная гистограмма имеет уже 3 оси измерений (на осях  $X$  и  $Y$  значения параметров, по которым распределяются события, на оси  $Z$  количество событий). Проблема для создателя некой гистограммы даже не в том, чтобы увидеть искомую корреляцию самому, а в том, чтобы в её наличии убедились другие сотрудники,

которые не просматривали эту гистограмму сотни раз в разных видах. Поэтому методика визуализации является довольно громоздким, но важным инструментарием для сотрудников в нашей области науки.

Отобразить на плоскости (на бумаге, на экране монитора) настоящее трехмерное изображение, естественно, невозможно, поэтому применяются либо проекции трех измерений наподобие изометрической (опции «**surf**» или «**lego**» метода **Draw()**), либо два измерения  $X$  и  $Y$  отображаются на плоскости, а отображение величин третьего измерения  $Z$  подменяется каким-нибудь достаточно информативным для человеческого глаза суррогатом (цветом области бина  $XY$ , или числом меток, помещаемых в эту область, и т.п.).

### Отображение третьего измерения цветом

Для отображения шкалы  $Z$  цветом можно применить метод **Draw("col")** или **Draw("colz")**, во втором случае справа от гистограммы отображается столбец, информирующий, какой цвет соответствует какой величине. Это соответствие, называемое палитрой, по умолчанию настроено не самым наглядным образом в текущих версиях *ROOT*. Для отображения на экране или для цветной печати бывает полезно настроить палитру для разложения диапазона величин  $Z$  по цветам радуги (**gStyle->SetPalette(1)**), что нагляднее, поскольку последовательность цветов в радуге многим привычна. Для подготовки гистограмм к черно-белой печати целесообразнее настроить палитру на отображение оттенков серого **gStyle->SetPalette(8)**, возможно число от 2 до 9.

Поскольку число отображаемых градаций оси  $Z$  получается ограниченным, для большей наглядности часто возникает нужда отобразить её в логарифмической шкале. В

ручном режиме для этого нужно кликнуть в свободном поле нужной канвы и в новом меню выбрать опцию *SetLogz*.

После того, как корреляция найдена, её количественную оценку можно провести, сворачивая 2D гистограмму в одномерные объекты. Для этого стоит потренироваться в работе с проекциями гистограммы вдоль осей (объекты *ProjectionX* и *ProjectionY*, [1] Тема «ROOT», ресурс «Свертка 2D гистограмм»). Для получения количественных характеристик эти свертки, как и другие одномерные гистограммы можно, например, *фитировать*.

## Фитирование гистограмм

Для фитирования гистограмм, то есть для поиска математических функций, описывающих содержимое этих гистограмм, в *ROOT* используется метод ***TH1::Fit()***. Вот сигнатура этого метода и пояснение его параметров:

```
void Fit(const char *fname, Option_t * option,  
Option_t * goption, Axis_t xxmin, Axis_t xxmax);
```

**fname** - имя фитирующей функции. Это должно быть имя одной из предопределённых в *ROOT* функций, или имя функции, определённой пользователем. Вот список имён предопределённых функций, которые могут использоваться в функции ***TH1::Fit()*** :

- **gaus** - гауссиан с 3-мя параметрами:  $f(x)=p0*\exp(-0.5*((x-p1)/p2)^2)$
- **expo** - экспонента с 2-мя параметрами:  
 $f(x)=\exp(p0+p1*x)$
- **polN** - полином степени *N*:  $f(x)=p0+p1*x+p2*x^2+...$
- **landau** - функция Ландау с двумя параметрами.

**option** это опции настройки процесса фитирования, **goption** опции отображения, а **xxmin**, **xxmax** задают границы области фитирования.

**Фитирование предопределённой функцией.** Чтобы



фитировать гистограмму предопределённой функцией, достаточно просто использовать функцию `Fit()`, передав ей в качестве первого аргумента имя функции: `hist.Fit("gaus");`. Для предопределённых функций нет необходимости устанавливать начальные значения параметров, это делается автоматически.

**Фитирование пользовательской функцией.** Чтобы отфитировать гистограмму своей функцией, нужно её создать (объект класса `TF1`) и вызвать `Fit()`, передав туда имя созданной функции. Есть 3 способа создать `TF1`: 1) Используя набор "C++ - подобных" выражений и операций, определённых для класса `TFormula`; 2) То же что и п.1, но с параметрами; 3) Используя вашу функцию (подпрограмму).

Пример создания *TF1* с помощью *TFormula*:  
`TF1 * f1 = new TF1("f1","sin(x)/x",0,10);` только что созданный объект `f1` класса `TF1` может быть использован для создания следующего объекта класса `TF1`:  
`TF1 * f2 = new TF1("f2","f2 * 2",0,10);`

Пример создания *TF1* с параметрами. Следующий способ - добавить параметры в выражение для функции:  
`TF1 * f1 = new TF1("f1","[0]*x*sin([1]*x)",0,10);`  
Номер параметра - в квадратных скобках. Чтобы задать начальные значения параметров явно, можно использовать функцию `SetParameter()`: `f1->SetParameter(0,10);`. Этим устанавливается начальное значение нулевого параметра равным 10. Можно так же установить допустимый диапазон изменения параметров: `f1->SetParLimits(0,2,10);`. Этим задается диапазон изменения параметра номер 0 во время фитирования только в пределах от 2 до 10.

Создание *TF1* с помощью пользовательской функции.  
Третий способ создать `TF1` - это определить функцию самостоятельно и передать её имя в конструктор `TF1`. Тип возвращаемого значения и список передаваемых параметров

должны в точности совпадать со следующим примером:  
`Double_t fitf(Double_t *x, Double_t *par);`  
 Здесь параметр `Double_t *x` это указатель на массив переменных; размерность массива равна числу переменных от которых зависит функция, для одномерной гистограммы используется только  $x[0]$ , для двумерной -  $x[0]$  и  $x[1]$  и так далее. А `Double_t *par` это указатель на массив параметров. Вот пример фитирования с использованием функции, определённой пользователем:

```
double ourFitFunction( double *x , double *
par){
    return par[0]*exp(-par[1]*x[0])
+par[2]*exp(-pow((par[3]-x[0])/par[4],2));
}
TF1* ff = new TF1("ff",ourFitFunction, 1.0,
10.0, 5);
ff->SetParameter(0,10);    // expo parameter
ff->SetParameter(1,0.5);    // expo parameter
ff->SetParameter(3,5);      // gaussian shift
ff->SetParLimits(4,0.1,1);  // gaussian sigma
hist->Fit("ff","q0r");
```

Сначала определяем функцию `ourFitFunction()` - она зависит только от одной переменной (используется только `x[0]`) и 5-ти параметров (используются `p[0]...p[4]`). Затем создаём объект `ff` класса `TF1` с именем `"ff"`, передаём ему указатель на `ourFitFunction()`, указываем диапазон функции (от 1.0 до 10.0) и число параметров (5).

## Практический пример фитирования

Теперь нужно отфитировать все 3 гистограммы. Можно это сделать в режиме *on-line*, кликнув на линию нужной гистограммы правой кнопкой мыши и выбрав опцию *ViewFitPanel*. После настройки процесса фитирования (выбор функции, флагов настройки, области фитирования) и нажатия кнопки ОК результаты фита

отобразятся в виде линии на графике. Полученные параметры функции фитирования будут отображены графически в легенде гистограммы, если была включена опция *SetOptFit* (кликнуть на гистограмму правой кнопкой, выбрать опцию, в появившемся поле 0 заменить на 1).

Можно этот процесс и автоматизировать. Для этого изменим скрипт, добавив туда код, доступный на сайте по адресу [1] (*Тема «ROOT», ресурс «Практика работы с гистограммами», параграф «Фитирование в скрипте»*). Ниже прокомментированы эти изменения:

**hist1->Fit("gaus","q0");** - фит гистограммы предопределённой функцией, причем заданные опции делают процесс фитирования проводимым без графического отображения линии функции и без текстового вывода в окно командной строки *ROOT*.

**TF1\* ff = new TF1("ff",ourFitFunction, 1, 10,5); ff->SetParameter(0,10); // expo parameter**  
- Создание *ROOT* функции для фитирования с областью определения от 1 до 10 и пятью параметрами и задание начального значения 10 нулевому параметру.

**hist1->GetFunction("gaus")->Draw("same");**

**hist1->GetFunction("gaus")->SetLineWidth(1);** - Нахождение объекта (функции «gaus») по его имени, и отображение его поверх гистограммы в нужный нам момент. Настройка толщины линии отображаемой функции.

После запуска скрипта, полученное изображение гистограмм с линиями функций и со значениями их параметров можно сравнить с образцом на сайте.