



Лекции. Практические занятия

Солдатов Е.Ю.

2024 г.

# ИСТОРИЯ

- Предшественник - пакет PAW реализован на языке **Fortran**. К середине 90-х этот язык программирования начинает устаревать.
  - В это же время набирает силу парадигма объектно-ориентированного программирования и язык программирования **C++**. Создание ускорителя **LHC** требует программного обеспечения, которое будет способно справиться с колоссальными объемами данных.
  - В 1995 году сотрудники **Европейской организации ядерных исследований (CERN)** Рене Бран и Фонс Ридмэйкерс выпускают первую версию программного пакета, реализованного на принципах ООП. Итогом этой работы становится программный пакет ROOT ***An Object Oriented Data Analysis Framework (Объектно-ориентированная среда для анализа данных)***
- 
- В 2003 году происходит, наконец, полный переход со старых фортрановских библиотек на пакет **ROOT**.
  - В настоящее время **ROOT** стал практически стандартом программного обеспечения для современных ускорительных экспериментов (далеко не только **CERN**).
  - На данный момент выпущена уже 6 версия данного пакета.
  - **ROOT** давно перерос **PAW** и содержит также множество разнообразных дополнений.

# ПОЛЕЗНЫЕ ССЫЛКИ

1. Сайт проекта: <https://root.cern.ch/>
2. Руководства пользователя к программе и её расширениям: <https://root.cern.ch/root-user-guides-and-manuals>
3. Справочное руководство (описание классов ROOT): <https://root.cern.ch/guides/reference-guide>
4. Раздел HowTo: <https://root.cern.ch/howtos>
5. Скачать ROOT: <https://root.cern.ch/downloading-root>
6. Форум: <https://root-forum.cern.ch/>



ROOT  
Data Analysis Framework

User's Guide

May 2018

- Preface
- 1 Introduction
  - 1.1 The ROOT forum
  - 1.2 Contact Information
  - 1.3 Conventions Used in This Book
  - 1.4 The Framework
  - 1.5 Installing ROOT
  - 1.6 The Organization of the ROOT Framework
  - 1.7 How to Find More Information

*Установка возможна на любую систему в том числе и Windows, но понадобится Visual Studio*

# ЧТО ТАКОЕ ROOT?

**ROOT** — пакет программ и библиотек, разработанный для использования в качестве платформы хранения, обработки и представления экспериментальных данных физики высоких энергий.

Основная первоначальная задача пакета **ROOT** – представление результатов измерений (визуализация) в удобном виде: гистограммы, графики, диаграммы, таблицы.

В реальности, пакет намного мощнее и позволяет производить обработку данных.

Также содержит:

- инструменты статистического анализа,
- линейной алгебры,
- средства фитирования,
- средства четырёхвекторных вычислений,
- инструменты многовариантного анализа данных, то есть использования нейронных сетей и многое другое.

*Поддерживает многопоточность и вычисления на кластерах.*

Написан на языке C++, в современных версиях есть поддержка Python, R, Java script и др.

BM@N  
Baryonic Matter of Nucleons

КТО ИСПОЛЬЗУЕТ?

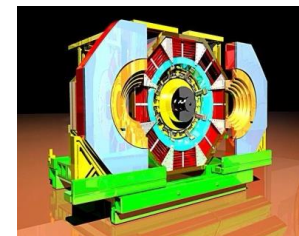


BABAR

DØ



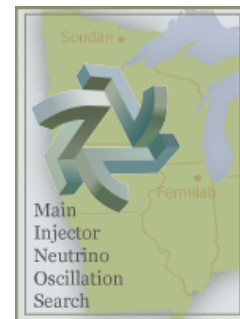
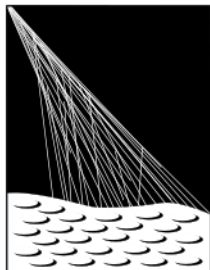
CRESST



MINERVA



PHENIX



NOvA



PIERRE AUGER OBSERVATORY



darkside

# СТРУКТУРА ROOT. ООП И КЛАССЫ ROOT

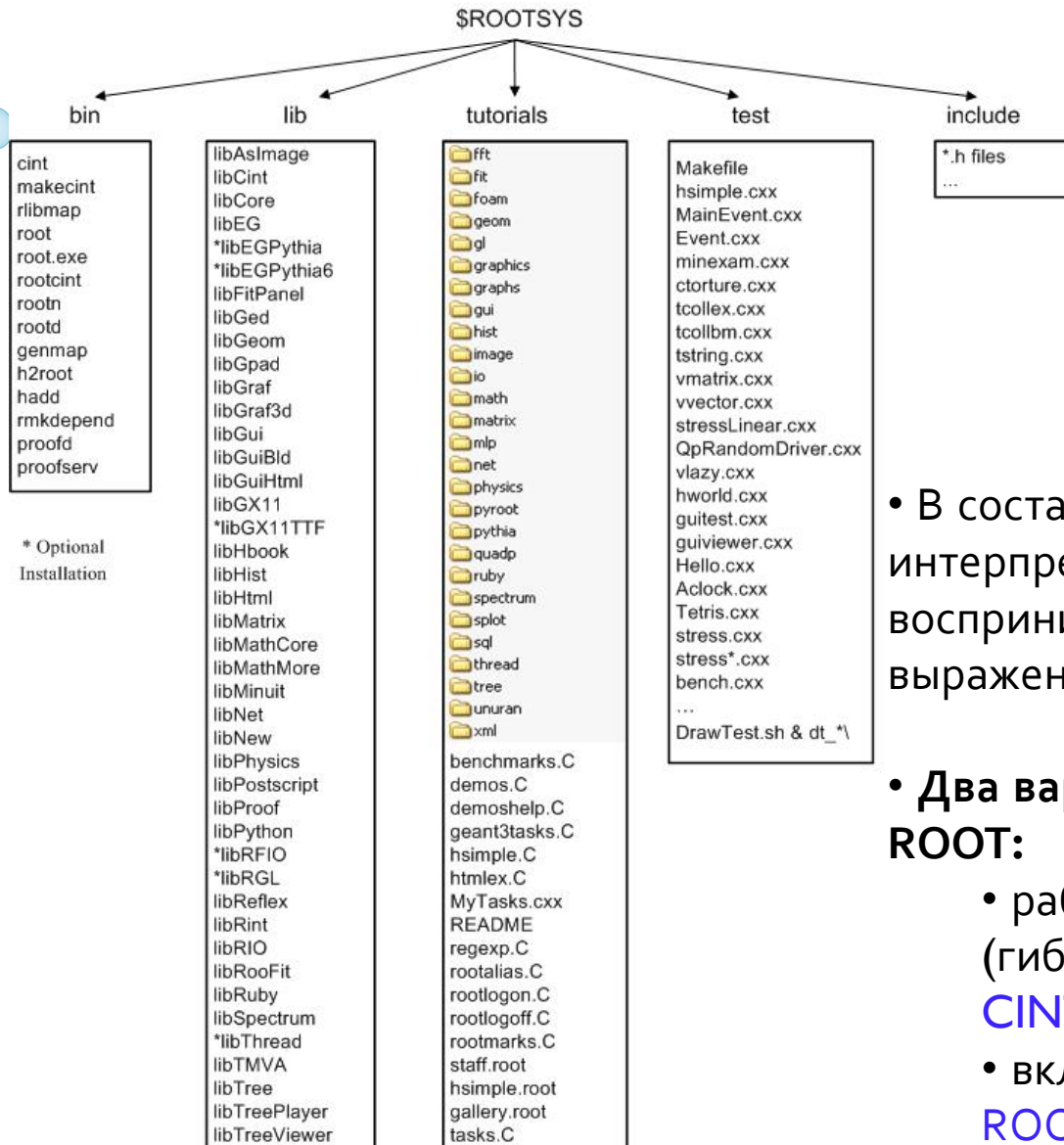
- Общая идея объектно-ориентированного программирования: моделирование окружающего мира как совокупности **объектов**, взаимодействующих друг с другом.
- Поддержка ООП в C++ реализуется с помощью **классов**
- **Класс** — это тип данных, определяемый пользователем
- **Класс** представляет собой модель реального **объекта** в виде данных и функций для работы с этими данными
- Функции класса называются **методами**, а данные — **полями**
- Принадлежность **метода** конкретному **классу** обозначается так: ***MyClass::DoSomething***
- **DoSomething** это **метод класса MyClass**
- **Объект** — это конкретный экземпляр, представитель данного **класса**

# СТРУКТУРА ROOT. УКАЗАТЕЛИ

- При работе с объектами часто используются указатели.
- Указателем называется переменная, в которой хранится адрес памяти, по которому располагается другая переменная
- Создание и определение указателя часто осуществляется с помощью операции `new`
- Создадим объект класса `MyClass` и указатель `pointer` на этот объект  
**`MyClass *pointer = new MyClass(...);`**
- Обращение к методам класса через указатель производится с помощью операции "`->`". Предположим, класс `MyClass` имеет метод `DoSomething(...)`
- Тогда обращение к этому методу через указатель `pointer` осуществляется следующим образом:  
**`pointer->DoSomething(...);`**

*Объекты, созданные с помощью операции `new` необходимо уничтожать с помощью `delete`.*

# СТРУКТУРА ROOT. УКАЗАТЕЛИ



- **ROOT** реализован как набор библиотек классов, обеспечивающих необходимую функциональность для работы с гистограммами, функциями, графиками, деревьями и т. д.

- В состав **ROOT** входит также интерпретатор **CINT/CLING**, который воспринимает команды **ROOT** и выражения **C/C++**

- Два варианта использования **ROOT**:

- работа в программе root.exe (гибкость благодаря **CINT/CLING** наподобие **Matlab**)
- включение библиотек классов **ROOT** в собственные программы



# ПРИНЯТЫЕ В ROOT ОБОЗНАЧЕНИЯ

- Имена классов начинаются с **T**  
*TFI, TTree,*
- Переменные типа «не класс» заканчиваются на **\_t**  
*Int\_t, Char\_t*
- Поля начинаются с **f**  
*fIntegral*
- Методы начинаются с прописной  
*Fill(), Draw(), Fit()*
- Константы начинаются с **k**  
*kRed, kBlue*
- Глобальные переменные начинаются с **g**  
*gStyle*

# МАШИННО-НЕЗАВИСИМЫЕ ТИПЫ ДАННЫХ В ROOT

В ROOT используются машинно-независимые типы данных, то есть их размер строго определён.

Наиболее употребляемые типы:

|                 |                              |         |
|-----------------|------------------------------|---------|
| <b>Char_t</b>   | Знаковый символьный          | 1 байт  |
| <b>Int_t</b>    | Знаковый целый               | 4 байта |
| <b>UInt_t</b>   | Беззнаковый целый            | 4 байта |
| <b>Float_t</b>  | Вещественный                 | 4 байта |
| <b>Double_t</b> | Вещественный                 | 8 байт  |
| <b>Bool_t</b>   | Логический (0=false, 1=true) |         |

Если вы не хотите сохранять переменную на диск, то можно использовать и `int` и `Int_t`, результат будет тот же. Интерпретатор и компилятор с ними будут работать совершенно одинаково.

# ROOT: НАЧАЛО РАБОТЫ

Чтобы запустить ROOT, наберите в консоли

`$ root`

версия

```
-----  
| Welcome to ROOT 6.24/02                                     https://root.cern |  
| (c) 1995-2021, The ROOT Team; conception: R. Brun, F. Rademakers |  
| Built for win32 on Jun 28 2021, 09:28:51 |  
| From tags/v6-24-02@v6-24-02 |  
| With MSVC 19.23.28107.0 |  
| Try '.help', '.demo', '.license', '.credits', '.quit'/'.'q' |  
-----  
  
root [0] |
```

↑  
Приглашение командной строки

Завершается сеанс работы командой `.q`

`root[.] .q`

`$ root --help`

- выдаст возможные ключи запуска

`$ root -l`

- быстрый запуск без приветственного окна

# СИНТАКСИС ИНТЕРПРЕТАТОРА CINT/CLING

CINT/CLING воспринимает 3 типа команд:

1. Команды самого **CINT/CLING** начинаются с «.»

|               |                                     |
|---------------|-------------------------------------|
| .?            | Вывести список возможных команд     |
| .L <filename> | Загрузить файл filename             |
| .x <filename> | Загрузить и выполнить файл filename |

2. C/C++ выражения в соответствии с синтаксисом языка

```
Int_t a = 12  
a++  
Int_t b=a*2
```

3. Команды **SHELL** начинаются с «.!»

```
.! ls  
.! pwd
```

# ИНТЕРАКТИВНЫЙ РЕЖИМ

Кроме вышеупомянутых команд интерпретатор сразу «понимает» продвинутые математические функции, которые находятся в пространстве имён TMath.

```
root [0] 2*2
(int) 4
root [1] sqrt(9.)
(double) 3.0000000
root [2] 3 > 4
(bool) false
root [3] TMath::Pi()
(double) 3.1415927
root [4] TMath::Sin(1.57)
(double) 0.99999968
```

Получается своеобразный «продвинутый» калькулятор.

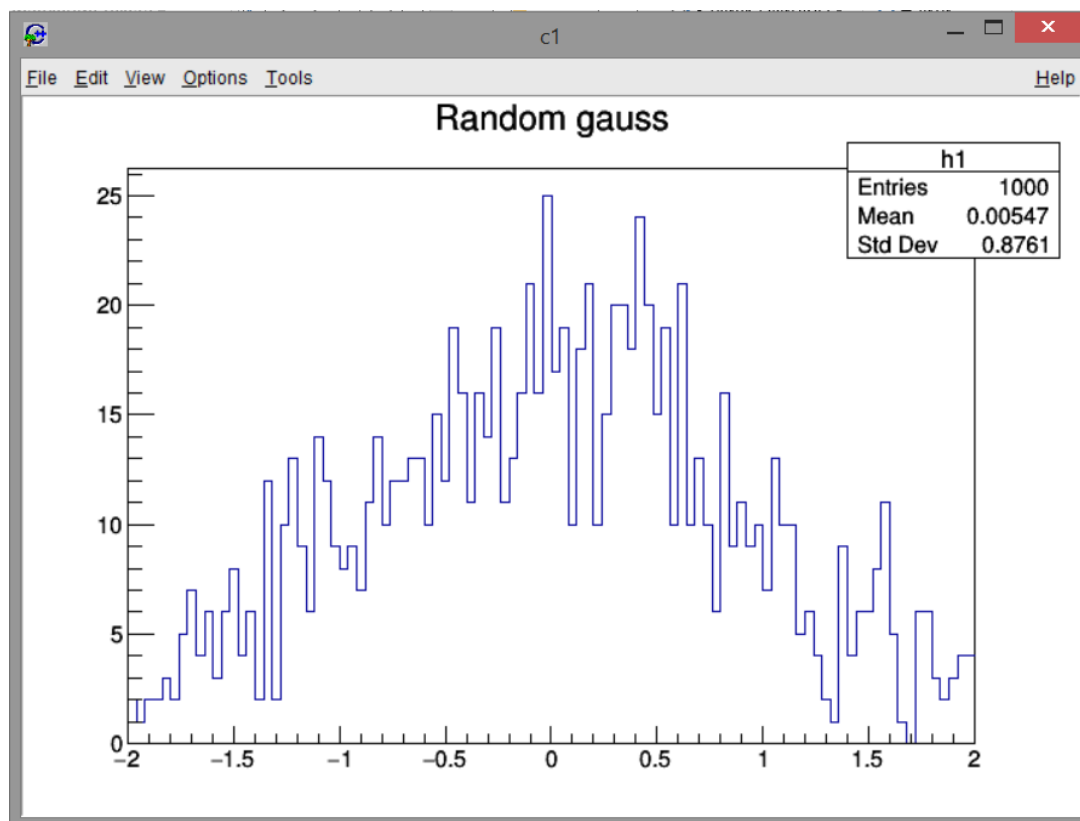
# ПРИМЕР ПРОСТЕЙШЕЙ СЕССИИ: ПОСТРОЕНИЕ ГИСТОГРАММЫ ЗНАЧЕНИЙ, ПОДЧИНЯЮЩИХСЯ НОРМАЛЬНОМУ ЗАКОНУ РАСПРЕДЕЛЕНИЯ (ГАУССА)

```
root [0] TH1F *h1 = new TH1F("h1", "Random gauss", 100, -2, 2)
```

```
root [1] h1->FillRandom("gaus", 1000)
```

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

Info in <TCanvas::MakeDefCanvas>: created default TCanvas with name c1



# ТИПЫ СКРИПТОВ: НЕИМЕНОВАННЫЕ

- В ROOT существует два типа скриптов: именованные и неименованные
- Неименованный скрипт представляет собой простую последовательность команд. Тело скрипта должно быть заключено в фигурные скобки

Пример простейшего скрипта. Файл `UnNamedMacro.C`

```
{  
#include <iostream>  
using namespace std;  
  
for (Int_t i=0; i<15; i++) {  
    cout<<i<<endl;  
}  
}
```

- Чтобы выполнить неименованный скрипт в интерактивной сессии  
`root [0] .x UnNamedMacro.C`
- ROOT будет искать скрипт в текущей директории, а также в директории `$ROOTSYS/macros`

Можно задать полный путь к файлу, например

```
root[0] .x /home/user501/UnNamedMacro.C
```

0  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14

# ТИПЫ СКРИПТОВ: ИМЕНОВАННЫЕ

- Чтобы иметь возможность включить в тело скрипта функцию, следует пользоваться именованными скриптами
- Пример именованного скрипта, содержащего функцию drawhist().
- Файл NamedMacro.C

```
void drawhist() {  
    TH1F *h1 = new TH1F("h1", "histogram", 10, 0, 10);  
    h1->Fill(1);  
    h1->Fill(3);  
    h1->Fill(5);  
}
```

- Чтобы выполнить функцию drawhist(), следует сначала загрузить скрипт в память ROOT, затем вызывать функцию

```
root[0] .L NamedMacro.C  
root[1] drawhist()
```



# ТИПЫ СКРИПТОВ: ИМЕНОВАННЫЕ

В именованный скрипт (макрос) возможно передавать переменные

```
void Example(int num, double weight)
{
    cout<<"Int number: "<<num<<" float weight: "<<weight<<endl;
}
```

```
root [0] .L Example.C
root [1] Example(5,0.99)
Int number: 5 float weight: 0.99
root [2]
```

Компиляция макроса с созданием общей библиотеки:

[.L Example.C+](#)

Перестройка библиотеки происходит только если макрос или другие файлы, которые он включает, новее чем библиотека.

Для принудительной перекомпиляции используйте «++»