

Пакет обработки данных ROOT

22.03.2016

Что такое ROOT?

- ▶ *ROOT is an object-oriented framework aimed at solving the data analysis challenges of high-energy physics.*

ROOT — объектно-ориентированная среда предназначенная для анализа данных в физике высоких энергий

<http://root.cern.ch/drupal/content/users-guide>

- ▶ *ROOT — пакет объектно-ориентированных программ и библиотек, разработанных в Европейском центре ядерных исследований. Пакет был разработан специально для использования в качестве платформы обработки экспериментальных данных физики высоких энергий и содержит специфичные для этой области продукты, однако также может быть использованы для анализа других данных, например, в астрономии.*

<http://ru.wikipedia.org/wiki/ROOT>

История и особенности

- ▶ 1986 год — PAW (fortran)
- ▶ 1995 год — ROOT (C++)
- ▶ Большие и сложные системы
 - ▶ Длинный жизненный цикл
 - ▶ Стабильность API
 - ▶ Переносимость
 - ▶ Независимость от сторонних библиотек
 - ▶ Эволюция
- ▶ Современный инструмент в ФВЭ (CERN, Fermilab...), астрофизике, ядерной физике и т.д.

Информация про ROOT

- ▶ Website ⇒ <http://root.cern.ch>
- ▶ Руководство пользователя ⇒ <http://root.cern.ch/drupal/content/users-guide>
- ▶ Справочное руководство ⇒ <http://root.cern.ch/drupal/content/reference-guide>
- ▶ Страница загрузки ⇒ <http://root.cern.ch/drupal/content/downloading-root>
- ▶ Примечания к выпуску ⇒ <http://root.cern.ch/root/html534/notes/release-notes.html>
- ▶ Примеры ⇒ [/tutorials](#)
- ▶ Тесты ⇒ [/test](#)

3 режима использования ROOT

1. Интерактивный ввод команд
2. Запуск файла-сценария для интерпретатора CINT/CLING
`$ root program.C`
3. Подключение ROOT к программе на Си++
`$ g++ -o prog prog.cpp 'root-config --cflags --libs'`

Интерпретатор команд CINT/CLING

```
source /opt/root/bin/thisroot.sh
root -l
x = 2.
y = 3.
z = x + y
.q
```

```
root
.x XXX.C
.q
```

Примеры графических возможностей

- ▶ OpenGL

- ▶ `tutorials/gl/gviz3d.C`
- ▶ `tutorials/gl/glsurfaces.C`
- ▶ `tutorials/gl/gh3c.C`
- ▶ `tutorials/gl/glbox.C`
- ▶ `tutorials/gl/glvox1.C`
- ▶ `tutorials/gl/glvox2.C`

- ▶ LaTeX и т.п.

- ▶ `tutorials/graphics/latex.C`
- ▶ `tutorials/graphics/feynman.C`

Полезные примеры

▶ Графики

- ▶ `tutorials/graphs/graph.C`
- ▶ `tutorials/graphs/gerrors.C`
- ▶ `tutorials/graphs/bent.C`
- ▶ `tutorials/graphs/multigraph.C`

▶ Гистограммы

- ▶ `tutorials/hist/fillrandom.C`
- ▶ `tutorials/hist/draw2dopt.C`
- ▶ `tutorials/hist/twoscales.C`
- ▶ `tutorials/hist/th2polyHoneycomb.C`
- ▶ `tutorials/hist/hsumTimer.C`

Полезные примеры

▶ Аппроксимация

- ▶ `tutorials/fit/fit1.C`
- ▶ `tutorials/fit/fit2dHist.C`
- ▶ `tutorials/fit/fitCircle.C`
- ▶ `tutorials/fit/fitExclude.C`
- ▶ `tutorials/fit/graph2dfit.C`
- ▶ `tutorials/fit/qa2.C`
- ▶ `tutorials/fit/myfit.C`
- ▶ `tutorials/fit/FittingDemo.C`
- ▶ `tutorials/hist/DynamicSlice.C`

Создание файла с данными

- ▶ Создать текстовый файл `data.dat`
- ▶ Заполнить строками вида:

0	1024
1	512
2	256
3	128

- ▶ Сохранить

Шаблон программы

- ▶ Создать текстовый файл prog.C
- ▶ Структура программы имеет следующий вид:

```
#include <TGraph.h>
...
double fitFcn(double* x, double* par){...}
...
void prog(){...}
```

- ▶ Запуск программы

```
root
.x prog.C или .x prog.C++
.q
```

Задание

Написать программу которая

- ▶ Читает созданный файл с данными

```
TGraph* g = new TGraph("data.dat");
```

- ▶ Выводит данные в консоль

```
for(int i = 0; i < n; i++){...
```

```
g->GetN();
```

```
double x, y;
```

```
g->GetPoint(i, x, y);
```

```
cout << x << "□" << y << endl;
```

- ▶ Строит график
- ▶ Фитирует подходящей функцией
- ▶ Выводит в консоль оценённые значения параметров зависимости

Задание

Необходимые шаги

- ▶ Отобразить график

```
g->Draw("A*");
```

- ▶ Задать свою функцию

```
TF1* f = new TF1("f","[0]*exp([1]*x)",0,4);  
f->SetParLimits(0, 100, 10000);  
f->SetParLimits(1, -10, 0);
```

- ▶ Профитировать своей функцией

```
g->Fit("f");
```

- ▶ Получить оценку значений параметров исследуемой зависимости