Introduction to Root program:L2

Presented by

DR. MOHAMMED ATTIA MAHMOUD

-PhD, Fayoum University, Egypt and Antwerp University, Belgium.

-Researcher in ENHEP, ASRT, Fayoum Uni, and BUE.

-FSQ Gen-Contact, CMS experiment, CERN, Geneva, Switzerland.

CINT Extensions to C++

1. Declaration can be omitted

```
f = new TFile("Example.root")
```

2. "." notation rather than "->"

```
f.ls()
```

3. Search for an object by its name

```
TH1F *smallHisto = new TH1F
                ("small","fPx 100",100,-5,5);
small->Draw();
```

Warning: These will not work in compiled code!

CINT Commands

- [expression] evaluates the expression

```
root[3] 3*4
(int)12
```

- .files show loaded source files
- .class [name] show class definition
- .g prints all objects in the root session
- .ls ls on current directory
- .pwd list the current directory, canvas, and style.

Opening and Inspecting ROOT Histograms Saved in Files

To inspect your ROOT histogram enter:

```
TFile    f("myHistogram.root");
```

This will loads the contents of the root file myHistogram.root into a temporary file in your ROOT session called "f".

To look at the contents of this file, enter:

```
f.ls();
```

You should see a response like:

```
TFile**   myHistogram.root      Created for you by RooTupleManager
TFile*    myHistogram.root      Created for you by RooTupleManager
KEY:      TH1F h1d1;1       MC reco abs mtm difference
KEY:      TH1F h1d2;1       Reco track momentum
KEY:      TH1F h1d3;1       Tracks per Event
KEY:      TH1F h1d4;1       Momentum
KEY:      TH1F h1d5;1       TagInspector Status
```

you can add entries to bins in the histogram and redraw the output

when you redraw the file:

```
h1d4.Fill(2,4);
h1d4.Draw();
```

Scripts Examples

- Un-named Script: hello.C

```
{
  cout << "Hello" << endl;
}
```

- Named Script:say.C

```
void say(char * what = "Hello")
{
  cout << what << endl;
}
```

- Executing the Named Script

```
root [3] .x say.C
Hello
root [4] .x say.C("Hi there")
Hi there
```

Functions and Fitting

- Function Objects (TF1)
  - Three constructors for TF1
  - User Defined Functions
- Fitting
  - Fit()
  - Fitting with a user defined function
  - Fitting subranges and combining functions
  - Demonstration of background and signal function

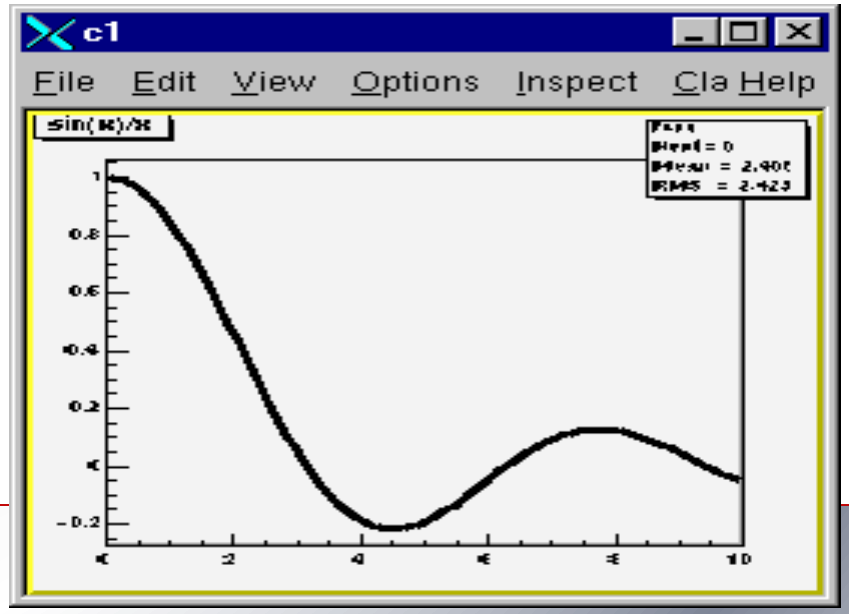
- Creating your own function objects
  - TF1, TF2, TF3
  - Three Signatures for the TF1 constructor

1. A C++ like expression using x with a fixed set of operators and functions defined in TFormula

```
TF1 *f1 = new TF1("f1","sin(x)/x",0,10);

f1->Draw();

TF1 *f2 = new TF1("f2","f1 * 2",0,10);
```



2. Same as the previous TF1 with Parameters

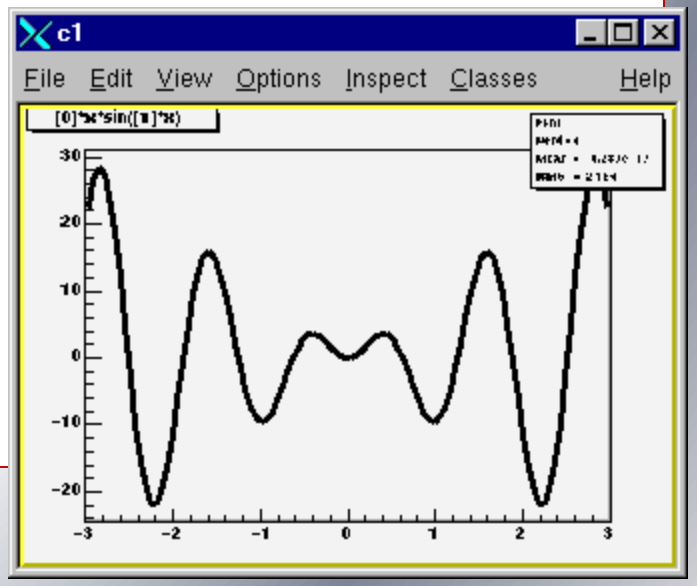
Call the constructor with parameter indices

```
TF1 *f1 = new TF1
   ("f1","[0] *x*sin( [1] *x)",-3,3);
```

See TFormula for valid expressions

Set the parameters explicitly

```
f1->SetParameter(0,10);
f1->SetParameter(1,5);
f1->Draw();
```




Thanks!