Corso di Laboratorio 2 – Programmazione C++

Silvia Arcelli

7 Novembre 2014

ROOT

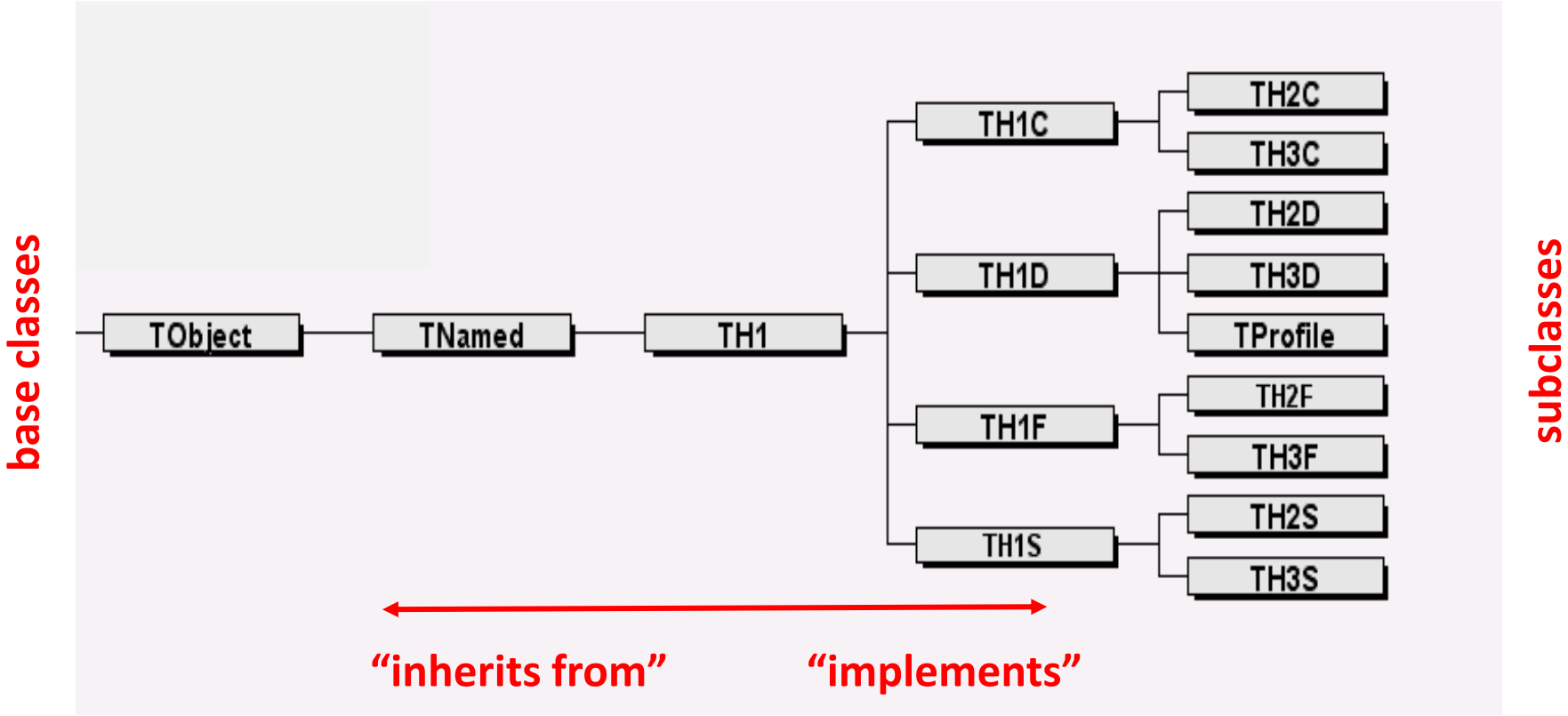
- Pacchetto software per l'analisi dati altamente versatile, che sonsente, fra le altre cose, di:
- Leggere/scrivere dati
- Selezionare dati secondo dei criteri
- Analizzarli (calcoli,fit)
- Risultati in forma grafica e numerica
- Salvare i risultati in diversi formati

ROOT

- Struttura Generale
- L'interfaccia Utente di ROOT
  - CINT (Command Line Interpreter, da ROOT 6 Cling)
  - Macros
  - Intefaccia Grafica (GUI)
- Classi per Istogrammi in ROOT

Gerarchia delle Classi

•Più di 1200 classi (il cui nome inizia sempre per T maiuscola) con funzionalità molto diverse. Implementazioni che utilizzano in maniera massiccia ereditarietà virtuale (anche multipla) e polimorfismo . Esempio per la classe di Istogrammi TH1:



Gerarchia delle Classi

- **oggetti della classe figlia sono anche del tipo della classe madre:**

  un istogramma TH1D „is a“ TObject

- **Le classi figlie hanno tutti i membri /metodi (pubblici e protected) delle classi madre:**

```
TH1D* hist=new TH1D("hpx","example",100,0,10);
cout <<"histogram name:" << hist->GetName()<<endl;
```

TH1D eredita la member function GetName() di TNamed

- **Le classi figlie eventualmente ridefiniscono i metodi dichiarati virtuali (e con il polimorfismo al run-time riconoscono correttamente il metodo da invocare):**

TObject::Print() "sovrascritto" da TH1::Print()

```
TObject* ob=new TH2I("map", "example", 50, 0, 1000,
50, 0, 1000);
ob->Print(); //automatically calls TH2I::Print();
```

TObject: "la" classe di base di ROOT

- **TObject definisce l'interfaccia dei metodi virtuali fondamentali:**

```
Draw(), Print(), Clear(), Streamer(), Clone(), Write(),...
```

- **Permette di definire il protocollo di l'IO (via TObject::Streamer()), ottimizzandolo (se volete salvare l'oggetto su un root file, dovete farlo ereditare da TObject) :**

```
ob->Write();
```

- **Tipo di base per l'uso delle root collections ("analoghi " ai container STL)**

```
TObject* ob= new TH1F("hpx","title",2048,0,2047);

 TList* list=new TList();
list->Add(ob);
TObject* ob2=list->FindObject("hpx");
```

- **runtime class introspection:**

```
TClass* cl=ob->Class(); // nome della classe,informazione sui
                        //  metodi e datamember della classe
if(cl->InheritsFrom("TH1"))... // check del tipo al runtime
```

ROOT Files

ROOT ha un'interfaccia di IO propria, implementata nella classe TFile,

- Un file ROOT può contenere oggetti e/o directories senza restrizioni.
- Un file ROOT file può essere manipolato in maniera interattiva (si possono aggiungere e cancellare oggetti, ad esempio). Si comporta esattamente come una qualunque altra directory in memoria
- Un file ROOT è di default compresso, in modo da usare uno spazio disco minimale

ROOT Global Variables

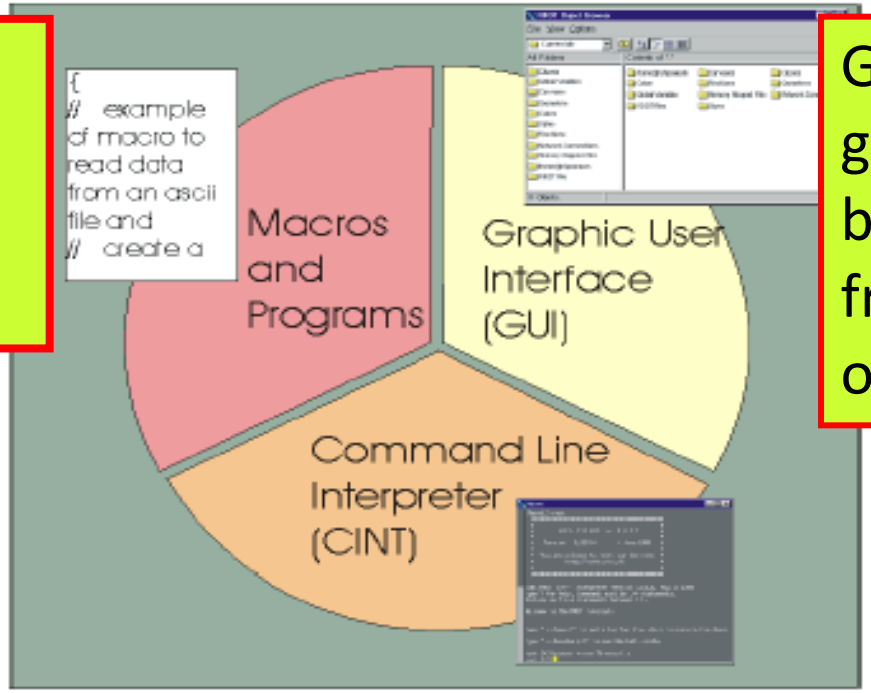
Una serie di variabili globali definite automaticamente durante la sessione di ROOT, sempre disponibili. Alcuni esempi:

- gROOT: informazione globale relativa alla sessione corrente attraverso il quale si può accedere praticamente a qualunque oggetto creato durante la sessione di ROOT
- gFile: puntatore al file corrente
- gDirectory: puntatore alla directory di root corrente
- gPad: puntatore alla pad corrente (finestre grafiche)
- gRandom: puntatore al generatore di numeri random

ROOT Interfaces

La User Interface: come l'utente può usare root

Uso di macro, codice compilato (CINT interpreter/ C++ compiler)

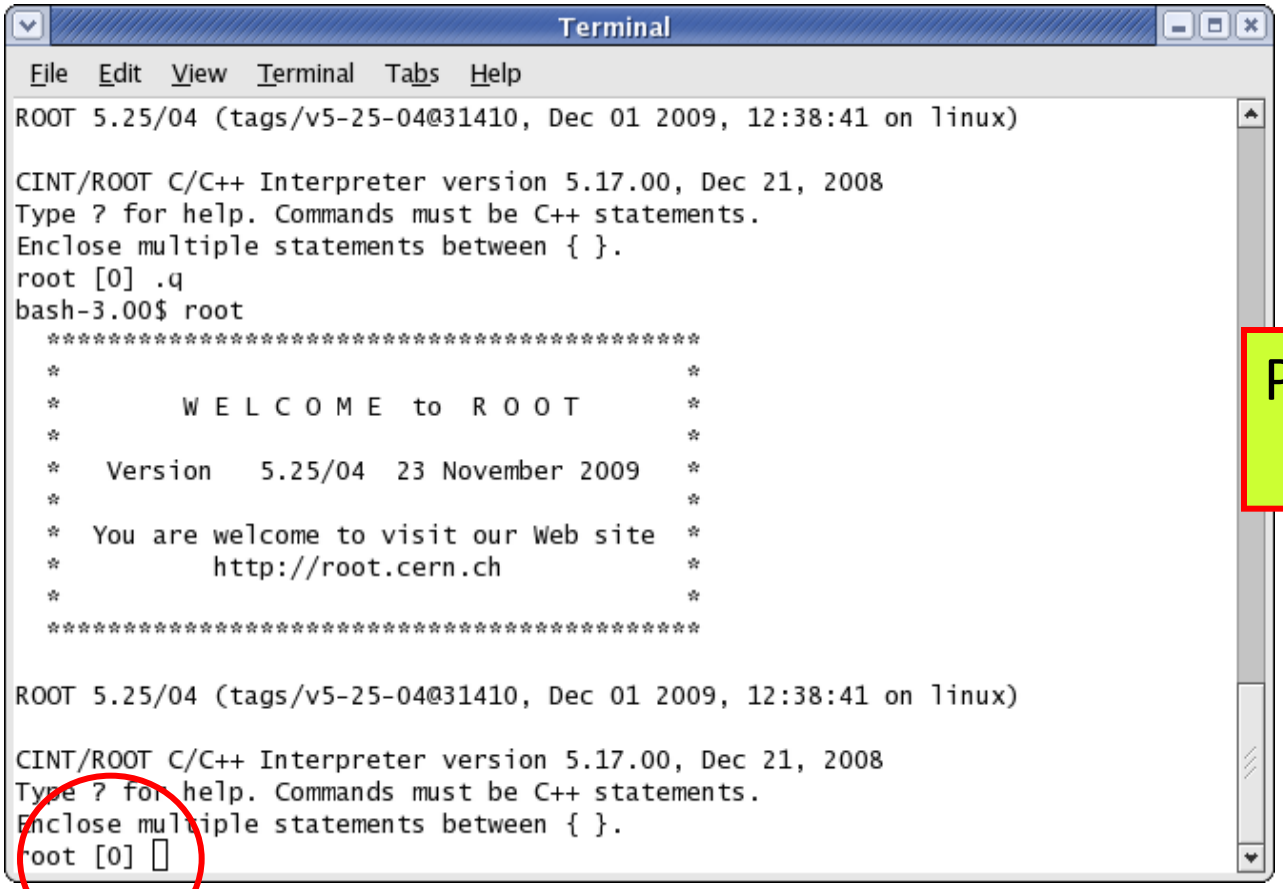


GUI: finestre grafiche, menu, bottoni. Più "user-friendly", utile per operazioni semplici

Attraverso CINT, con comandi diretti (C++ interpreter). Utile per operazioni semplici

ROOT Interfaces

La command-line User Interface:



Per far partire root:
>root

"prompt" CINT root

ROOT-CINT

Due tipi di comandi:

- **Comandi di base di CINT Iniziano tutti con il punto, “.”**

  Esempio: root[0] .q vi fa uscire da root

  Comandi di SHELL iniziano tutti con “.!”

  Esempio: root[0] .! ls vi fa un listing della directory corrente

- **Comandi con sintassi di tipo C++**

  Esempio: root[0] TBrowser b; apre una finestra grafica

  root[1] int a=3; definisce e assegna ad a 3

  root[2] a senza il ; stampa il tipo e il valore

  (int) 3 di a (fa da calcolatrice tascabile)

(il “;” in command line non è obbligatorio!)

ROOT-CINT

- Alcuni comandi di base di CINT:
- .q (.qqq, .qqqqqq) per uscire da root
- .files fa vedere le librerie/sorgenti caricate
- .L carica in memoria i simboli definiti in una macro
- .x carica ed esegue una macro
- .ls list della directory corrente in root
- .pwd mostra la directory, canvas, e style
- .? Lista/help sui comandi disponibili

ROOT-CINT

- Per comandi C++ in CINT, sintassi standard con alcune “libertà”:
  - Il ; può essere omesso su comando singolo
    - TBrowser *b= new TBrowser() // è ok!
  - Il tipo nelle dichiarazioni può essere omesso
    - f= new TFile(“example.root”);
  - Le notazioni per oggetti e puntatori a oggetti sono equivalentemente utilizzabili:
    - f.ls() equivalente a f->ls()
  - Accesso ad oggetti attraverso il nome, non solo attraverso il loro puntatore:
    - TH1F *histo=new TH1F(“histname”,” Titolo”, 100, 0, 10)
    - histname->Draw() equivalente a hist->Draw()

ROOT-CINT

root possiede dei tipi propri (machine independent) per i tipi nativi (portabilità del codice). Supporta comunque anche i tipi convenzionali del C++.

C++ type	Size (bytes)	ROOT types	Size (bytes)	FORTRAN analog
(unsigned)char	1	(U)Char_t	1	CHARACTER*1
(unsigned)short (int)	2	(U)Short_t	2	INTEGER*2
(unsigned)int	2 or 4	(U)Int_t	4	INTEGER*4
(unsigned)long (int)	4 or 8	(U)Long_t	8	INTEGER*8
float	4	Float_t	4	REAL*4
double	8 (=4)	Double_t	8	REAL*8
long double	16 (= double)			REAL*16

ROOT-CINT

- **CINT con comandi C++ in blocchi. Uso delle parentesi graffe:**

```
root [] {
end with '}'> Int_t j = 0;
end with '}'> for (Int_t i = 0; i < 3; i++)
end with '}'> {
end with '}'> j= j + i;
end with '}'> cout <<"i = " <<i<<", j = " <<j<<endl;
end with '}'> }
end with '}'> }
i = 0, j = 1
i = 1, j = 2
i = 2, j = 3
```

N.B. L'uso del ";" è qui obbligatorio

ROOT-CINT

- **Command-Line Help:**

  Molto utile l'utilizzo del <Tab> della tastiera:

  ad esempio per aprire un TBrowser:

  root[0] b=new TB<Tab>; vi dà una prima lista di match compatibili

  root[0] b=new TBrow<Tab> raffinate la ricerca...

  root[0] b=new TBrowser(<Tab> vi dà una lista dei possibili costruttori della classe

  root[0] TBrowser::<Tab> vi dà una lista dei metodi della classe

- **Per richiamare i comandi**, usare le frecce sulla tastiera

- La "storia" della sessione di ROOT è salvata nel file **.root_hist** nella vostra home

ROOT-MACRO

User Interface attraverso script (macro) e codice compilato:

- Unnamed script
  - Inserire tutto il codice che si vuole eseguire tra { }
  - Non possono essere dichiarate classi o funzioni
  - Non si possono usare parametri

- Named script
  - Come una qualunque funzione C++, stesse regole
  - Si possono definire funzioni e classi, usare parametri
  - La funzione che ha lo stesso nome del file viene eseguita con il comando CINT ".x nomefile"

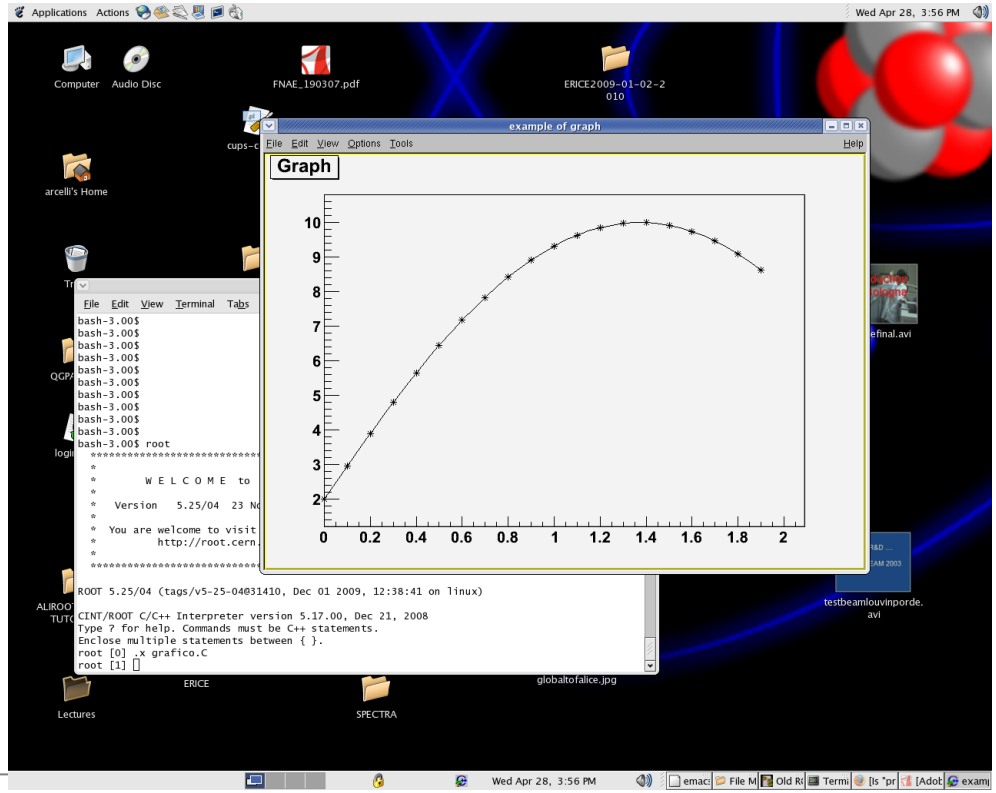
ROOT-MACRO

Esempio Unnamed Script:

Nel file grafico.C:

```
{
TCanvas *c1=new TCanvas("c1","Esempio di grafico",200,10,700,500);
const Int_t n=20;
Float_t x[n],y[n];
for(Int_t i=0;i<n;i++){
  x[i]=i*0.1;
  y[i]=10*sin(x[i]+0.2);
}
TGraph *graph=new TGraph(n,x,y);
graph->Draw("AC*");
}
```

root[] .x grafico.C



ROOT-MACRO

Esempio Named Script:

Nel file grafico.C:

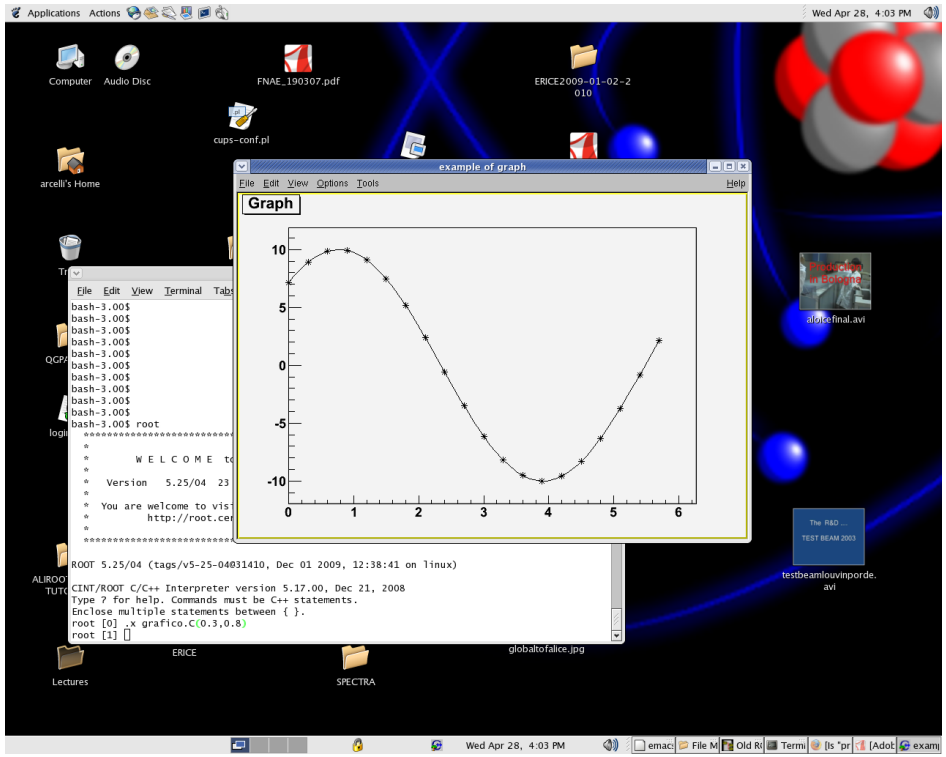
```
void grafico(Float_t scale,Float_t offset){
TCanvas *c1=new TCanvas("c1","Esempio di grafico",200,10,700,500);
Float_t x[n],y[n];
for(Int_t i=0;i<n;i++){
  x[i]=i*scale;
  y[i]=10*sin(x[i]+offset);
}
TGraph*graph=new TGraph(n,x,y);
graph->Draw("AC*");
}
```

root[] .x grafico.C(0.3,0.8)

oppure:

root[] .L grafico.C

root[] grafico(0.3,0.8)

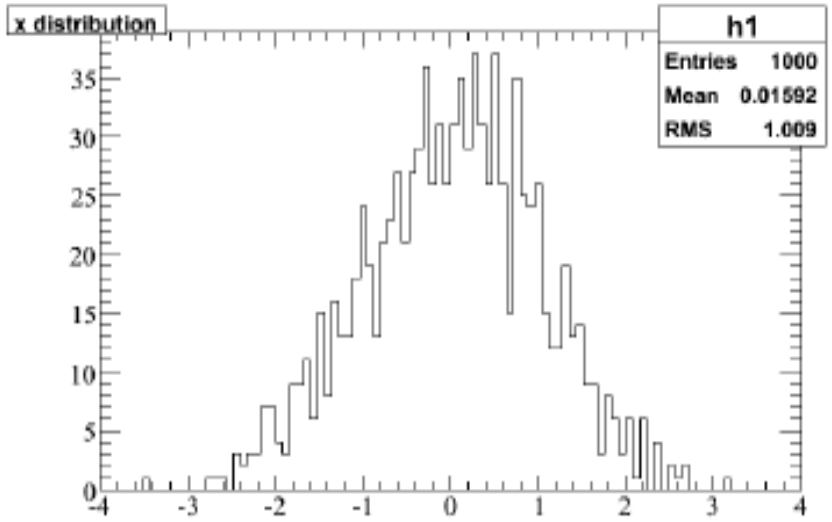


Istogrammi 1D : derivate di TH1

```
TH1F *name = new TH1F(”name","Title”,
  nBins, lowest bin, highest bin);
```

Esempio:

```
//dichiazione istogramma
TH1F *h1 = new TH1F("h1","x distribution",100,-4,4);
for(Int_t i=0;i<1000;i++){
//...
h1->Fill(x); //filling
}
h1->Draw();//drawing
```



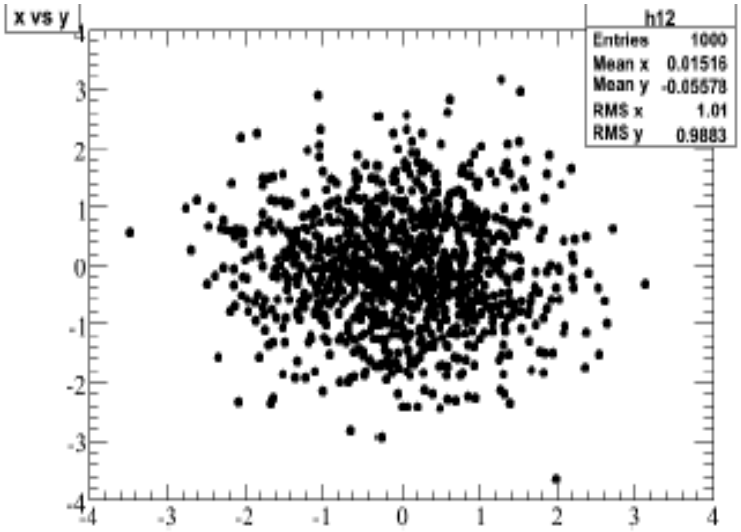
(Possibile anche binning
con larghezza dei bin variabile)

Istogrammi 2-D: TH2

```
TH2F *name = new TH2F(”name","Title”, xBins,
  low xbin, up xbin, yBins, low ybin, up
  ybin);
```

Esempio:

```
TH2F *h12 = new TH2F("h12","x vs y",100,-4,4,100,-4,4);
//...
h12->Fill(x,y); //filling, 2 variabili
//...
h12->Draw();
```

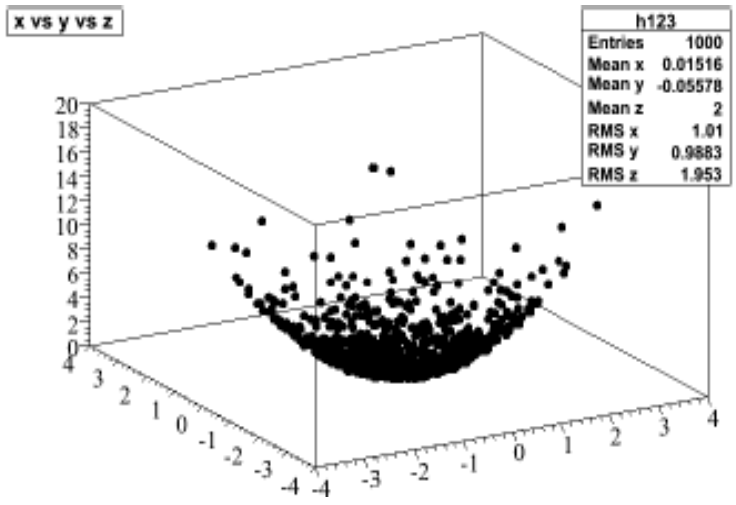


Istogrammi 3-D: TH3

```
TH3F *name = new TH3F(”name","Title”,
    xBins, low xbin, up xbin, yBins, low
    ybin, up ybin, zBins, low zbin, up
    zbin);
```

Esempio:

```
TH3F *h123 = new TH3F("h123","x vs y vs z",100,-
    4,4,100, -4, 4,100,0,20);
    h123->Fill(x,y,z);
    h123->Draw();
```

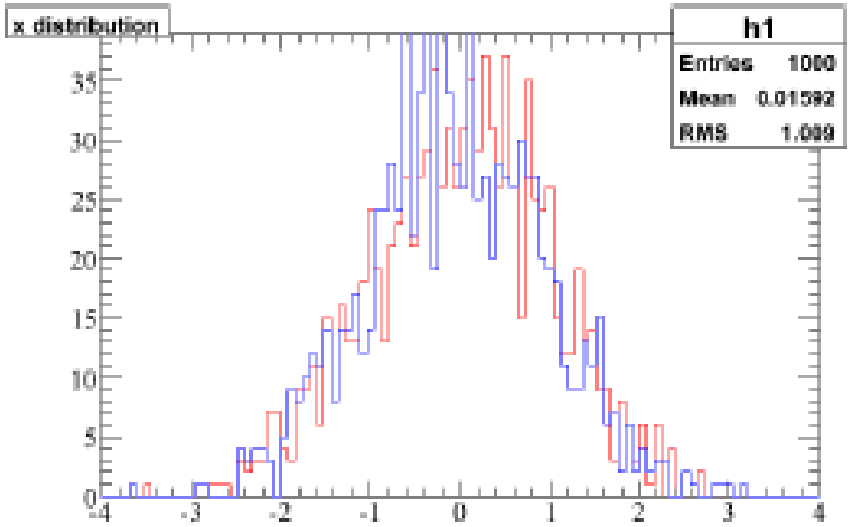


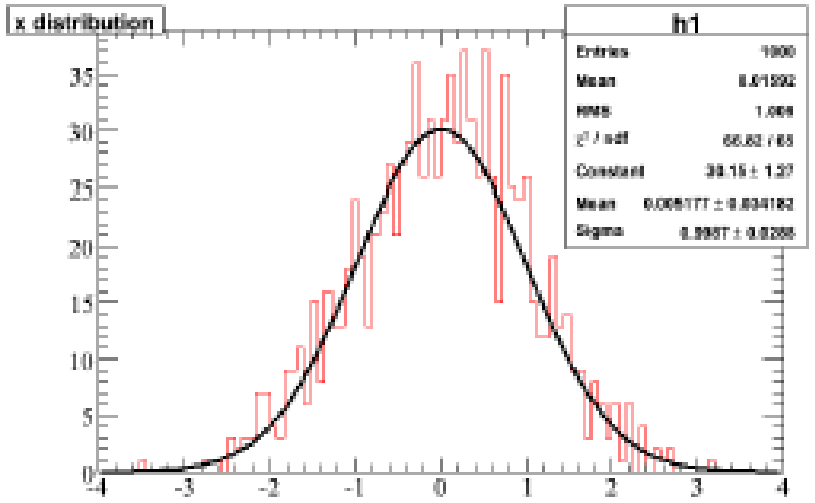
Per istogrammi N-dimensionali con N>3 usare THNSparse

Istogrammi 1-D: TH1

Sovrapporre istogrammi:

```
h1->Draw();
h2->Draw("same");
```





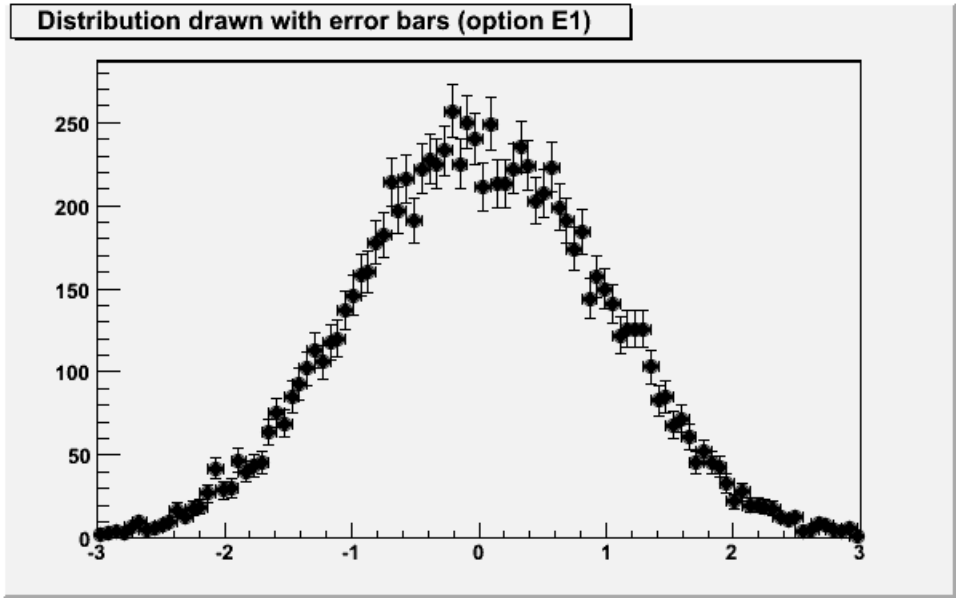
Fare un Fit:

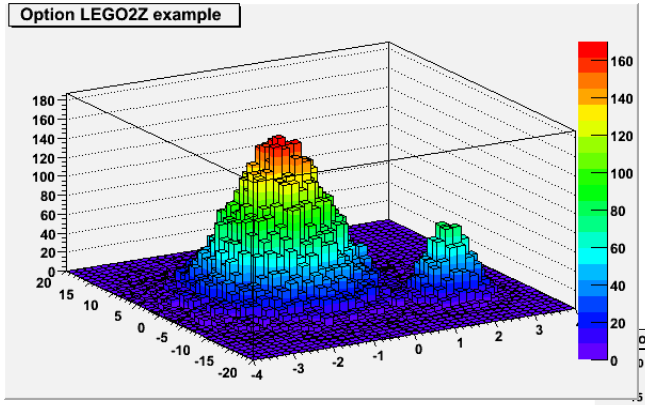
```
h1->Fit("gaus");
```

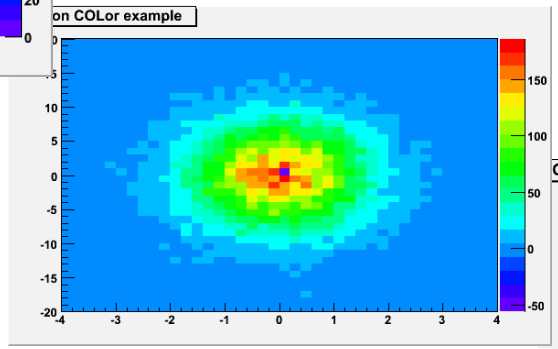
Istogrammi- Drawing Options

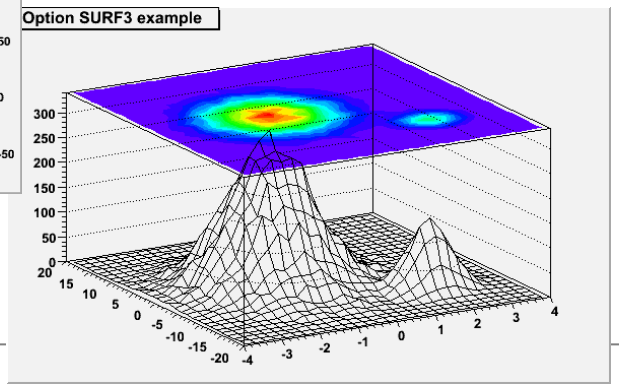
Alcune opzioni del metodo Draw():

- “E”: Mostra gli Errori
- “HIST”: disegna solo l’istogramma





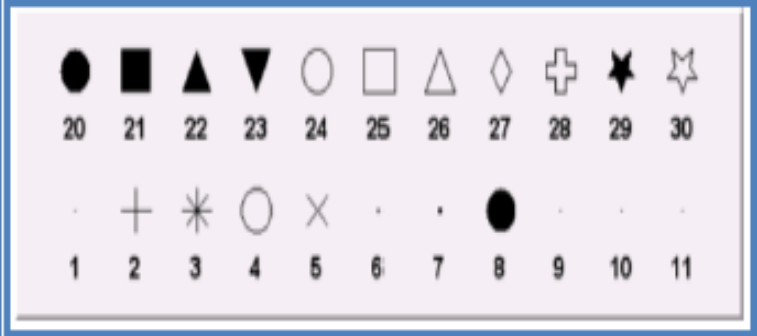




- "LEGO”
- “CONT”
- "SURF":

Istogrammi-grafica

h1.SetMarkerStyle();



h1.SetFillColor();

LineStyle
h1->SetLineStyle();

LineColor
h1.SetLineColor();

Alcuni Metodi degli Istogrammi

Command	Parameters
h1.GetMean()	Mean
h1.GetRMS()	Root of Variance
h1.GetMaximum();	Maximum bin content
h1.GetMaximumBin();	location of maximum
h1.GetBinCenter(int bin_number);	Center of bin
h1.GetBinContent(int bin_number);	Content of bin

Alcuni Metodi degli Istogrammi

- Somma, Sottrazione, Moltiplicazione, Divisione:

  su oggetti:

  ```
  TH1F h2=3*h1;
  TH2F h3=h1*h2;
  ```

  con puntatori:

  ```
  TH1F h2= 3*(*h1);
  ```

  con i metodi della classe:

  ```
  h->Add(...),h->Multiply(...),h->Divide(...),
  h->Scale(k)
  ```

- Integrali: `Integral()`, `ComputeIntegral()`, `GetIntegral()`
- Numeri Casuali: `FillRandom(funcname,N)` **riempe con N numeri random generati secondo funcname**, `GetRandom(x)` **genera un numero random distribuito secondo la pdf definita dall'istogramma**
- Proiezioni (per istogrammi >1-D)

Istogrammi,Esempio

Esempio: istogramma di una variabile generata secondo una distribuzione gaussiana

```
void histo(Float_t mean, Float_t width, Int_t nev)
{

  TFile *file = new TFile("example.root","recreate"); //apro il file root
  TH1F *h = new TH1F("histo", "example histogram ", 100, 0., 3); // creo istogramma

  Float_t x;
  for(Int_t i=0;i<nev;i++){
    x=gRandom->Gaus(mean,width); //genero random secondo una gaussiana
    h->Fill(x); //riempo l'istogramma
  }
  file->cd(); // mi posiziono sul file
  file->Write(); //scrive tutti gli istogrammi in memoria su file
  file->Close(); //chiudo il file
}
```

- Opzioni di apertura file:"new", "recreate","read",...
- Istogramma: nome, titolo, numero di bin, range
- gRandom->Gaus(x,y), pdf gaussiana predefinita in root

Istogrammi,Esempio

- Per poi accedere all'istogramma da root, da linea di comando:

```
root[] TFile *file =new TFile("example.root");
```

apro il file

```
root[] file->ls();
```

listing del contenuto del file (se non ci si ricorda come si chiama l'istogramma)

```
root[] TH1F *histo=file->Get("histo");
```

estraggo l'istogramma attraverso il suo nome

- Attraverso comandi C++ diretti o su macro potete manipolare il vostro oggetto (proprietà, operazioni, grafica)

- Oppure, potete utilizzare l'interfaccia grafica, che vi permette di usare gran parte dei metodi di base degli oggetti di root

Esempi

Macro che crea e popola tre istogrammi, e li scrive su un file ROOT:

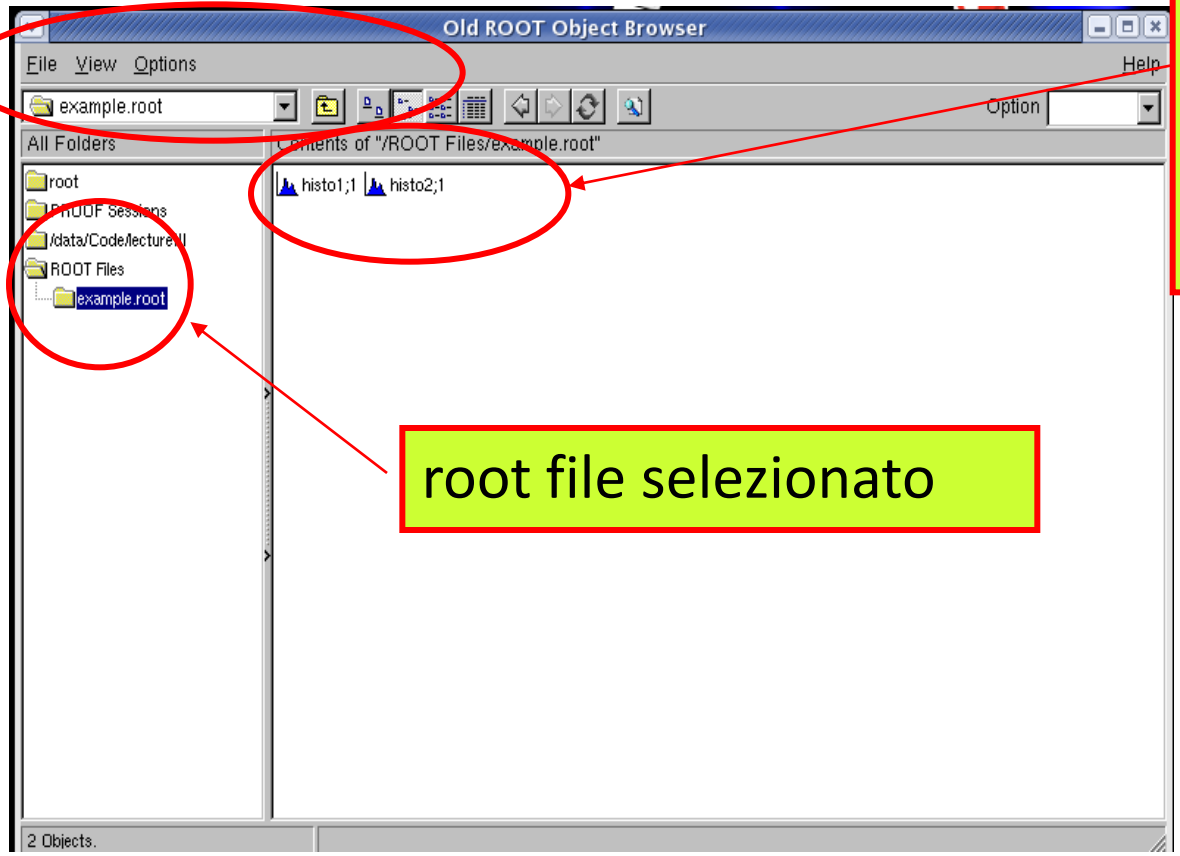
makeHistos.C

Macro che apre il file ROOT, legge gli istogrammi, e li disegna :

drawHistos.C

ROOT-GUI

Menus,toolbars



root file selezionato

Contenuto root file:

- **Subdirectory**
- **Oggetti root (istogrammi,canvas,..)**
- **trees**

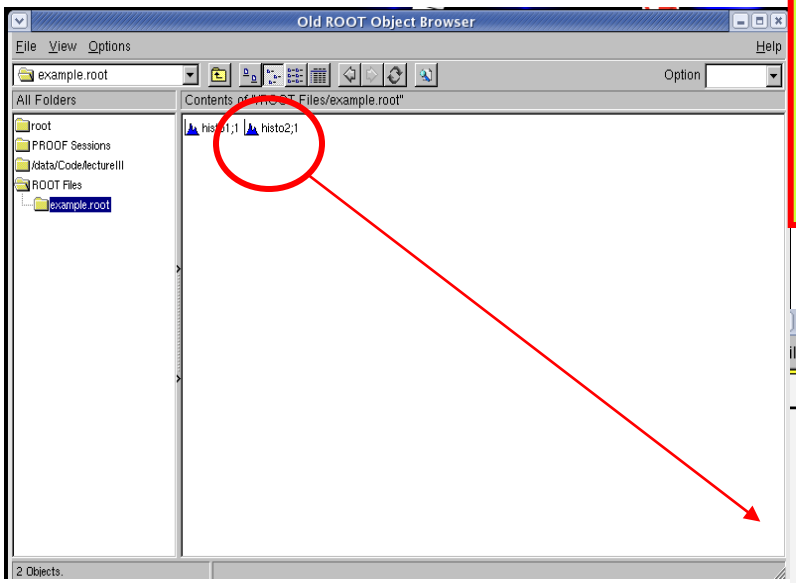
TBrowser:

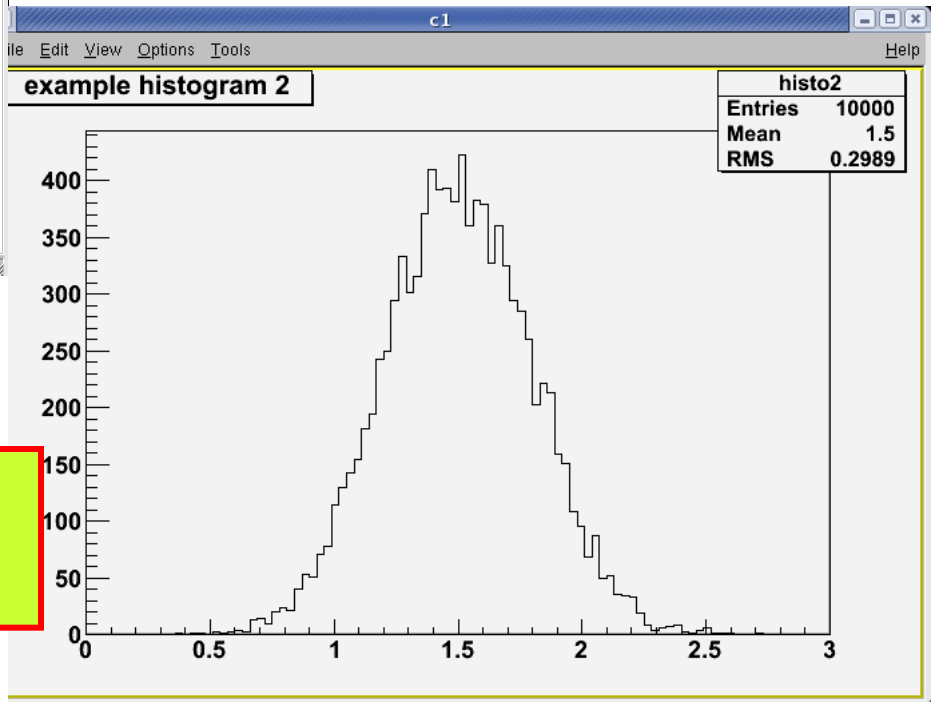
Finestra grafica per accesso a file di root

da root[] dare il comando TBrowser b;

ROOT-GUI

Clikkando 2 volte con il bottone sinistro del mouse su un oggetto nel file si apre un TCanvas, la finestra di interfaccia per la grafica

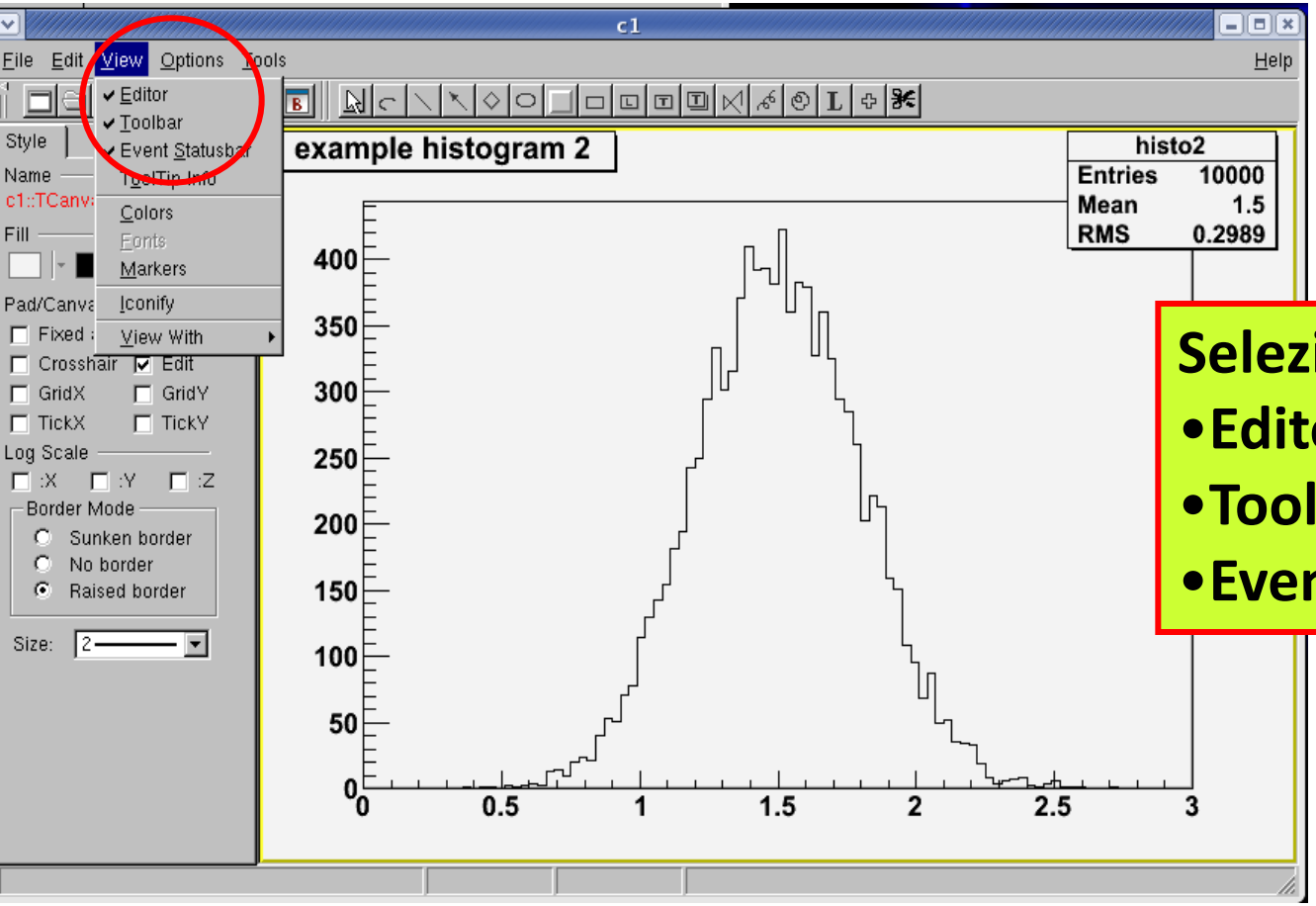




Opzioni grafiche di base, a questo livello...

ROOT-GUI

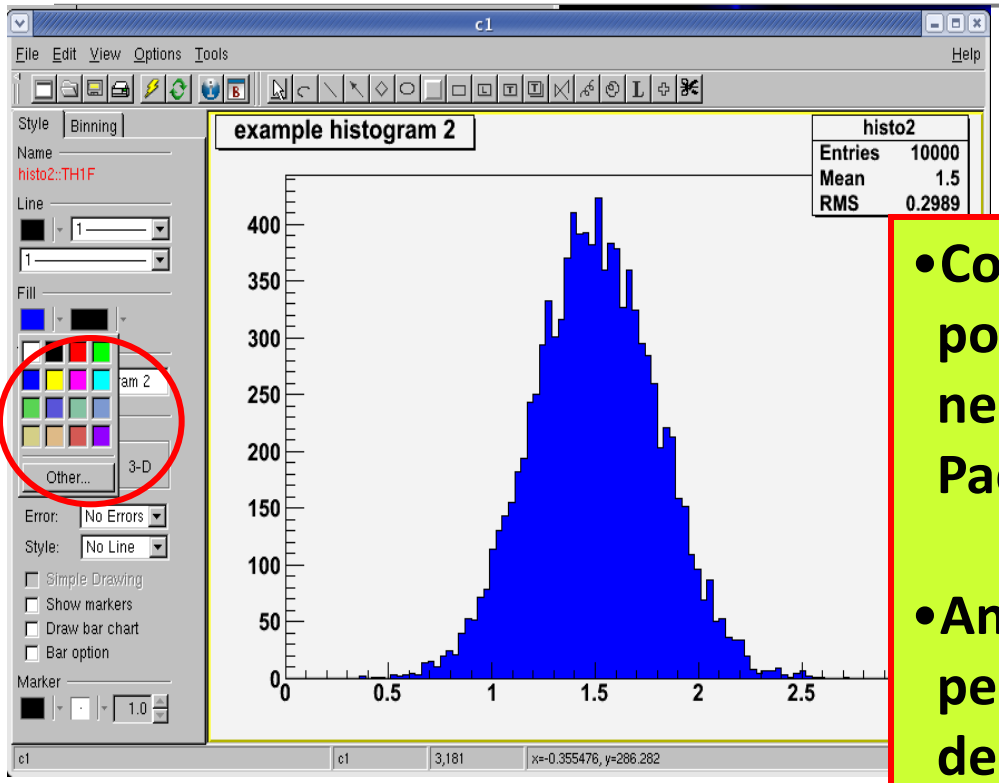
Come agire facilmente sulla canvas:



Selezionare in View:

- **Editor**
- **ToolBar**
- **Event StatusBar**

ROOT-GUI



Editor Menu

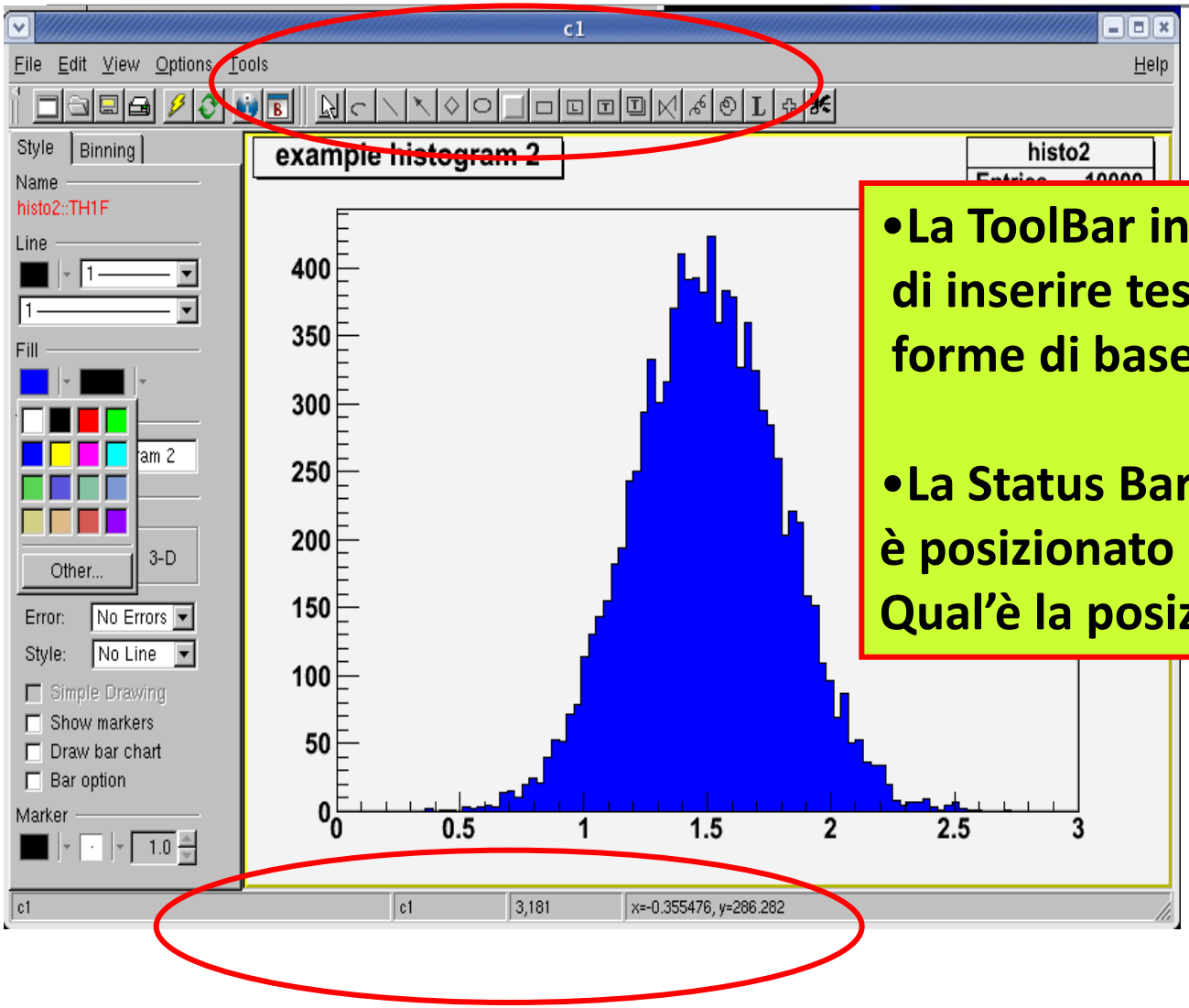
- Con il **bottone sx** del mouse potete **selezionare** i vari oggetti nella canvas (1 click): Pad, Frame, assi,istogramma,..

- Andate sull'editor a sinistra per **cambiare** i parametri dell'oggetto. L'editor cambia a seconda dell'oggetto selezionato

- Con il bottone sinistro potete anche spostare/ridimensionare gli oggetti (opzione SetEditable on)

ROOT-GUI

Toolbar, Status Bar

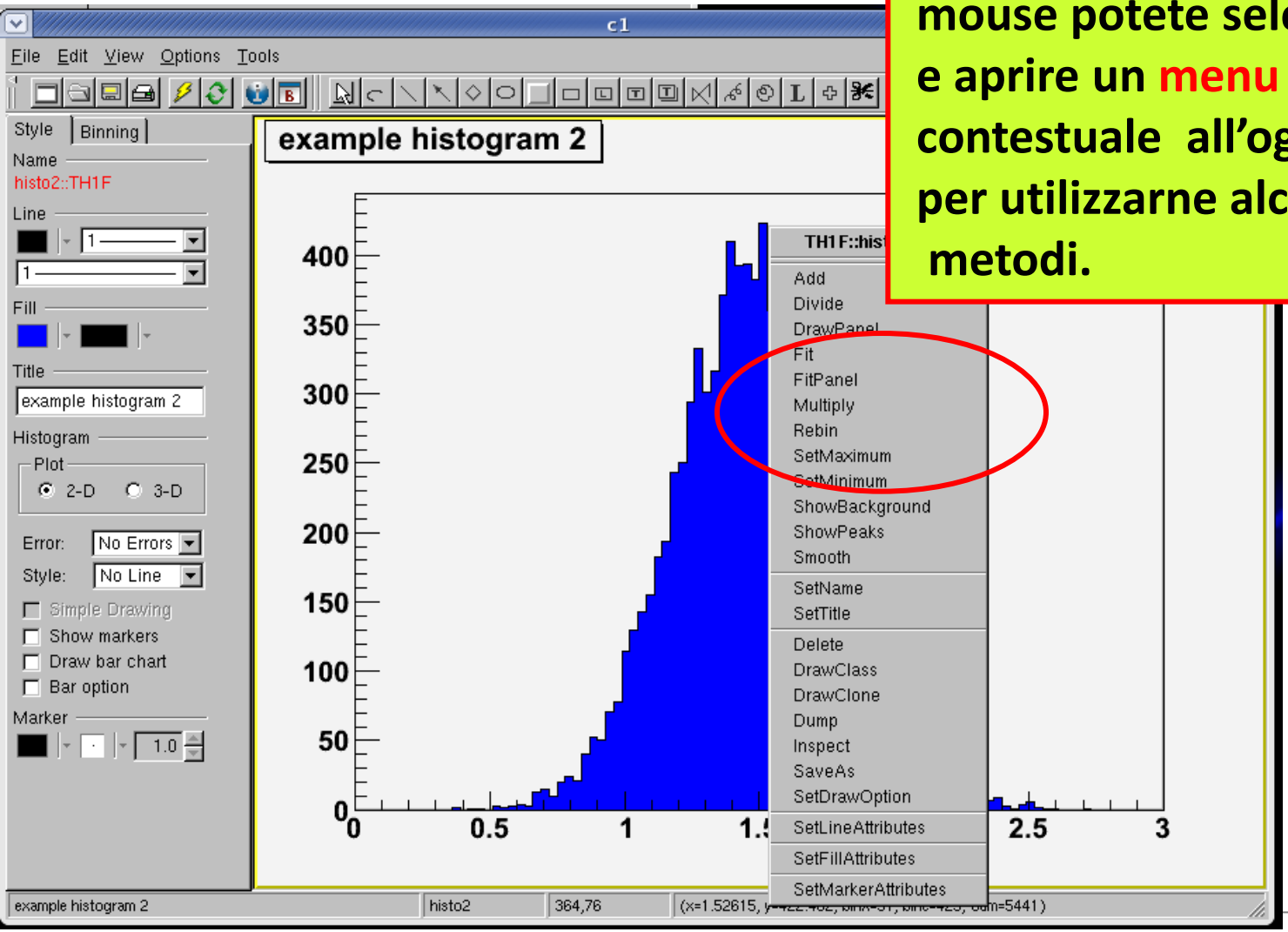


- La ToolBar in alto vi consente di inserire testo, simboli, frecce, forme di base

- La Status Bar vi dice su quale oggetto è posizionato il mouse e Qual'è la posizione sulla Canvas

ROOT-GUI

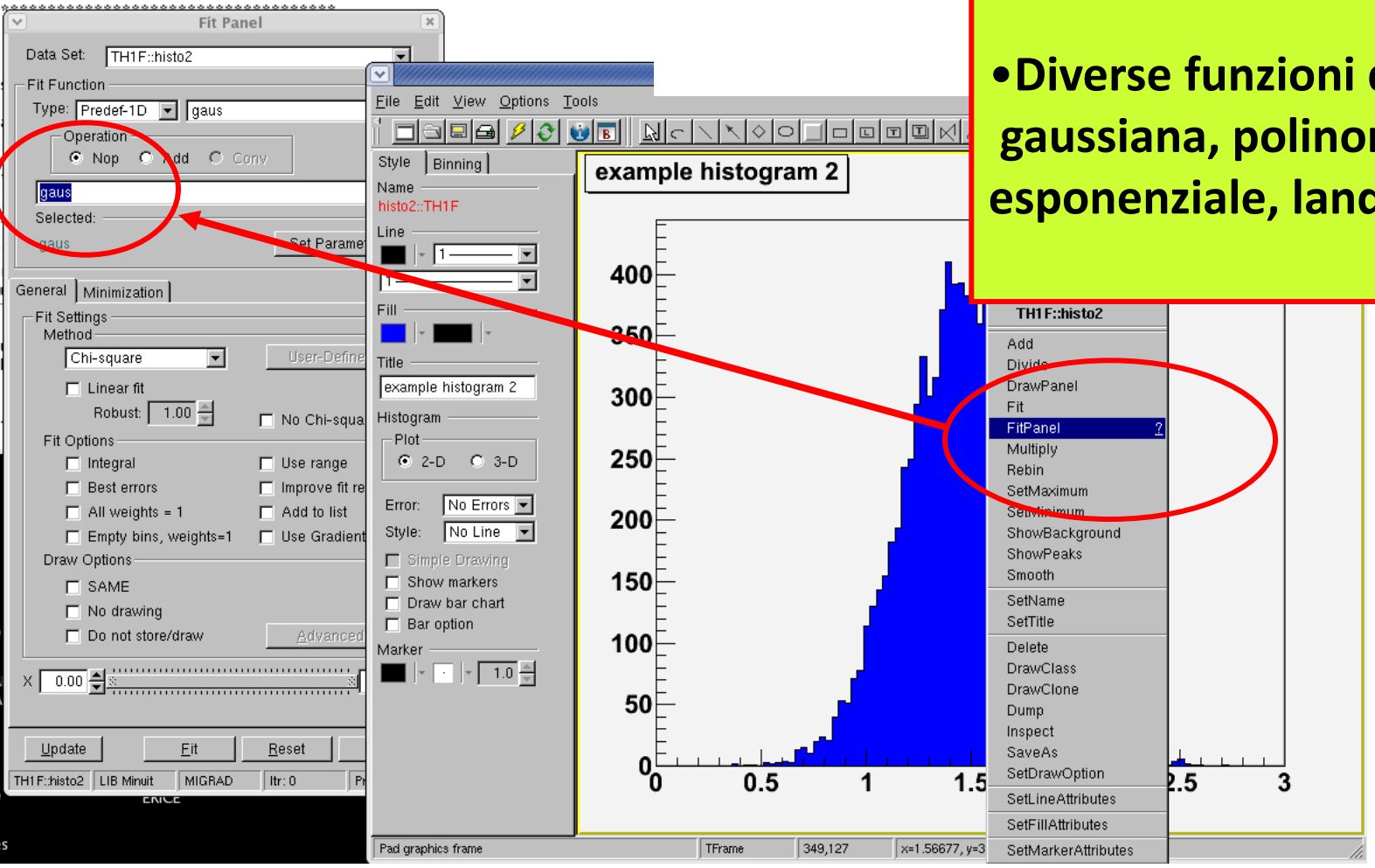
Con il **bottone dx** del mouse potete selezionare e aprire un **menu** contestuale all'oggetto, per utilizzarne alcuni metodi.



ROOT-GUI

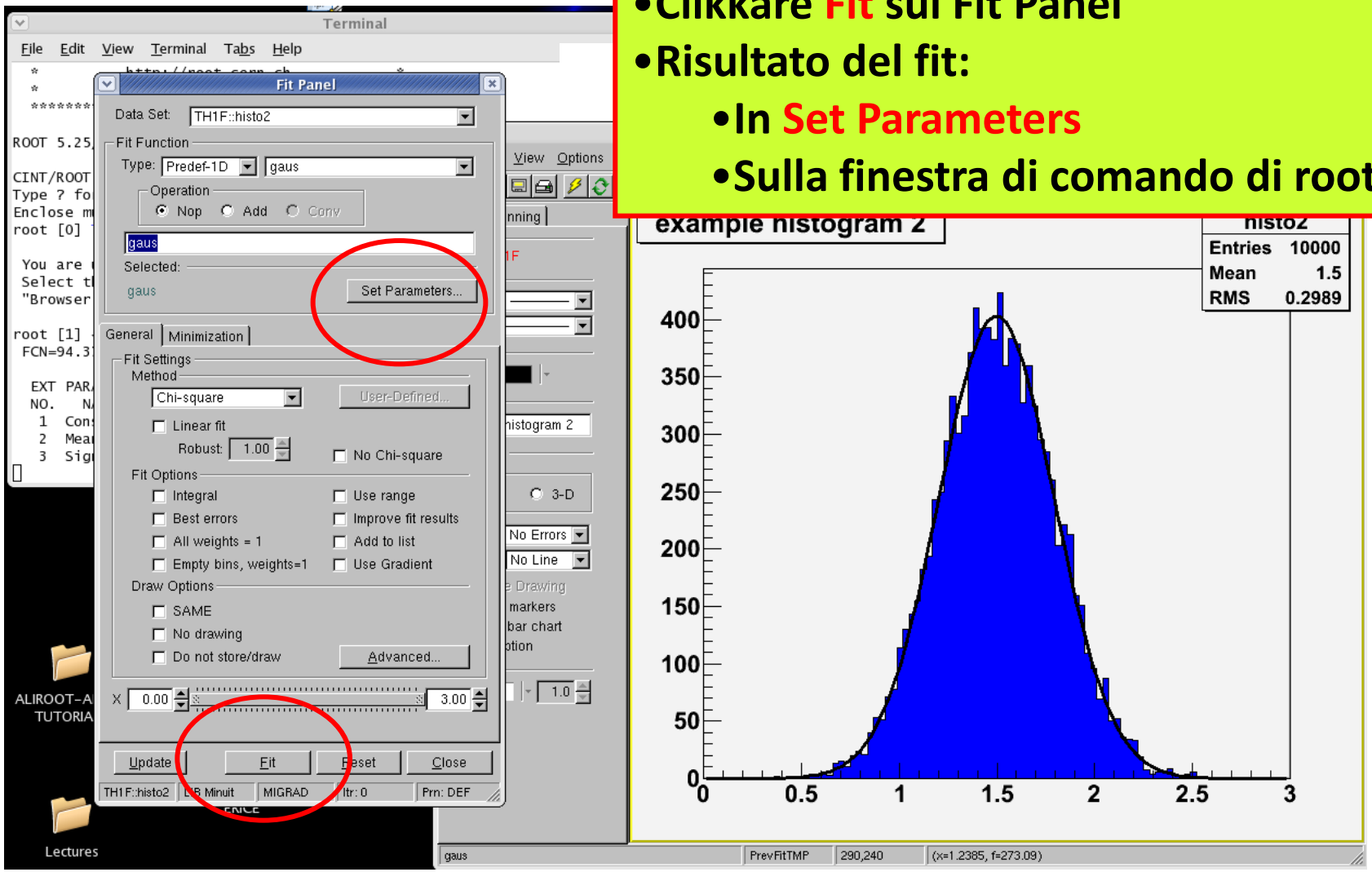
- Ad esempio, se vogliamo fare un fit dell'istogramma, selezioniamo "FitPanel":

- Diverse funzioni di default: gaussiana, polinomiale, esponenziale, landau

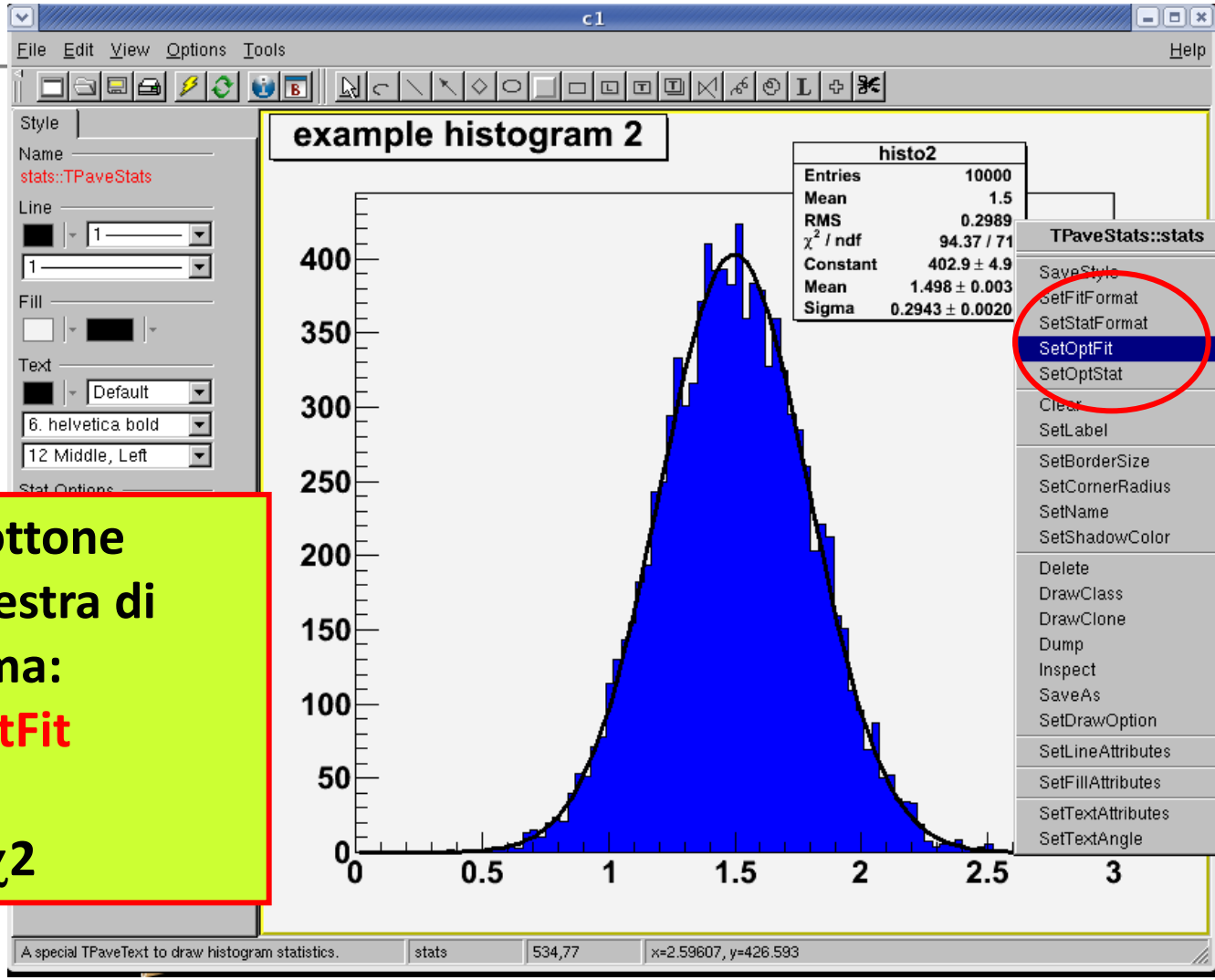


ROOT-GUI

- Clikkare **Fit** sul Fit Panel
- Risultato del fit:
  - In **Set Parameters**
  - Sulla finestra di comando di root



ROOT-GUI



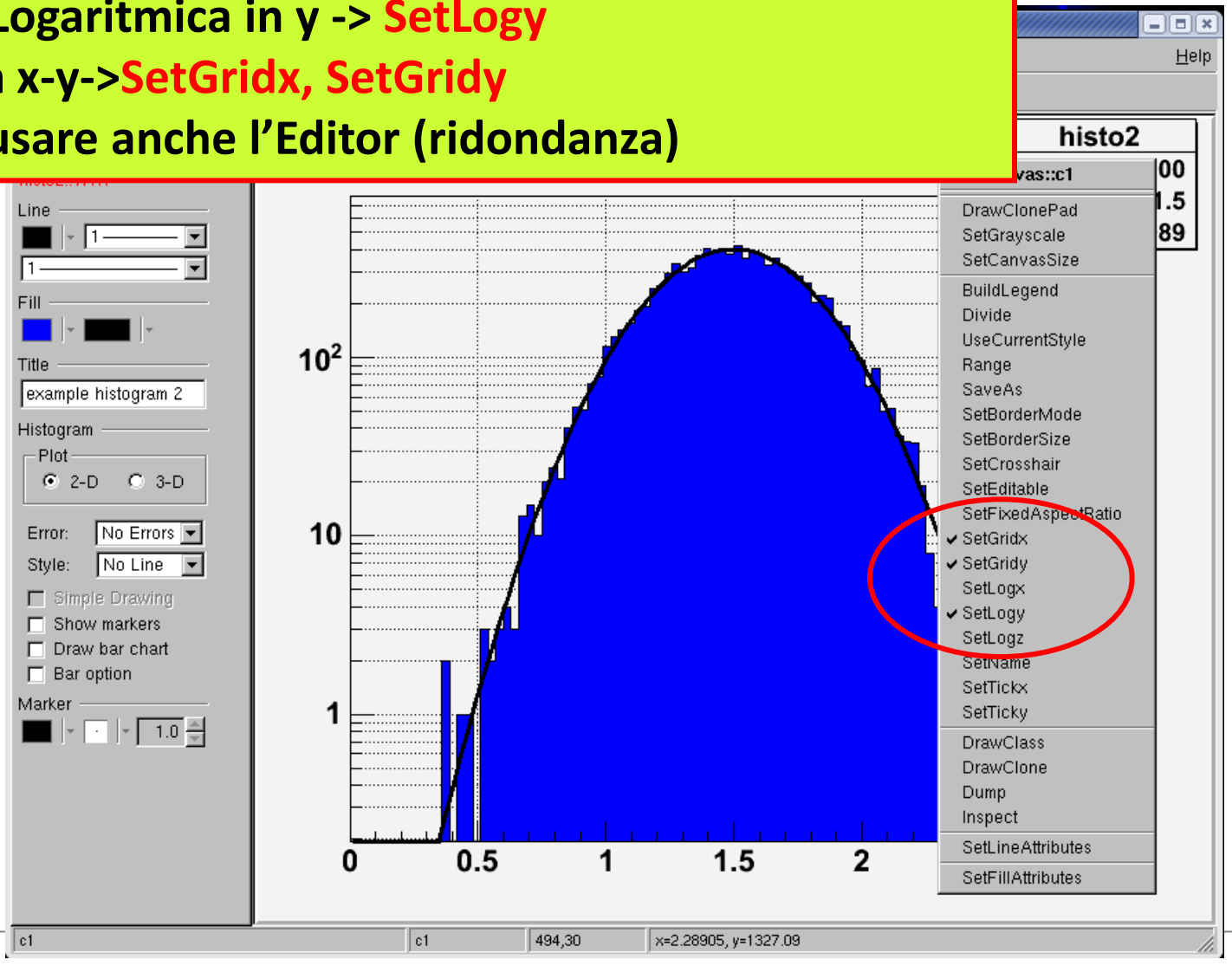
Oppure, aprire con il bottone destro il menu della finestra di statistica dell'istogramma:

- **Selezionare SetOptFit**
- **Scrivere 111**

Parametri con errori e $\chi 2$

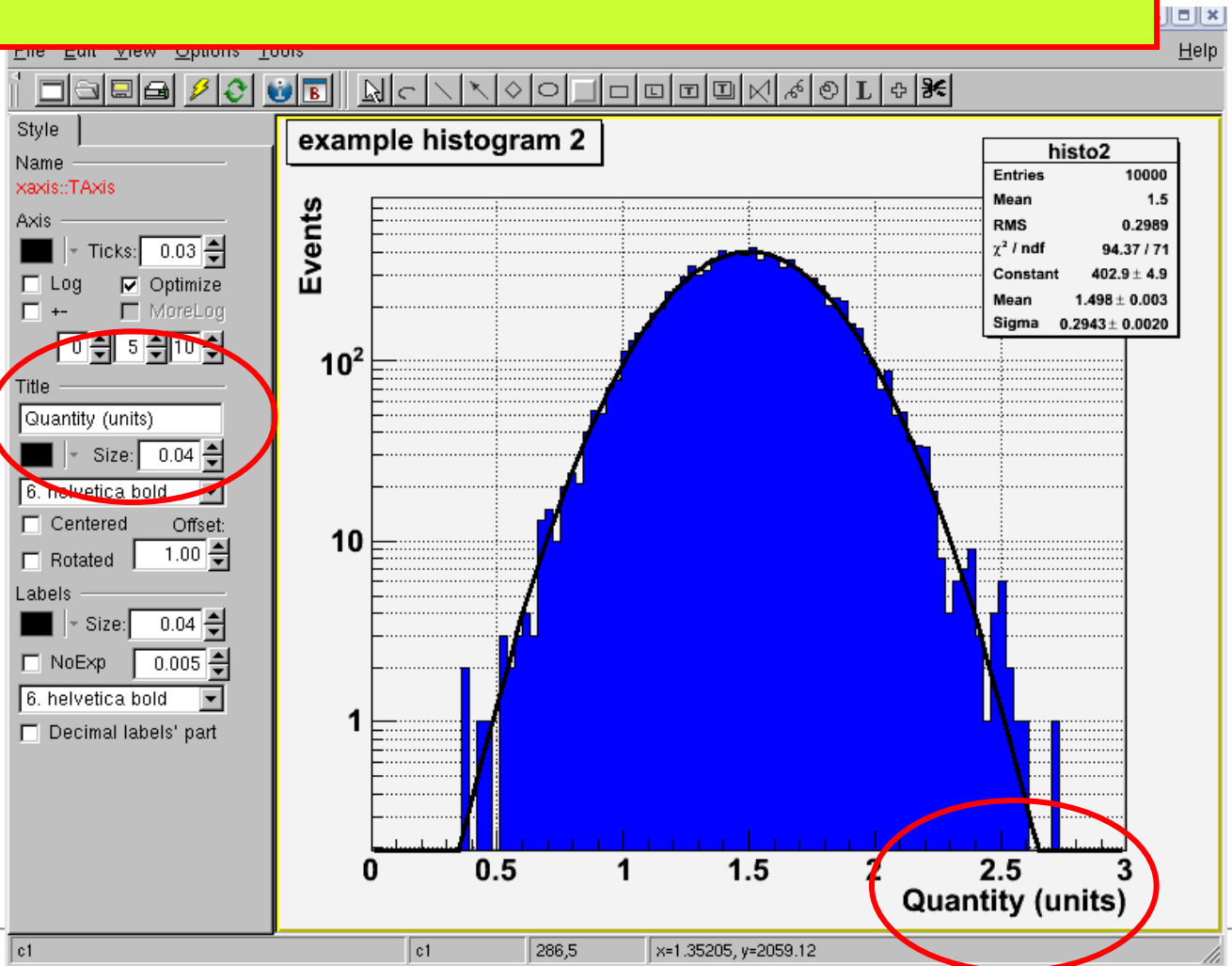
ROOT-GUI

- Altro esempio:uso del bottone dx sulla canvas:
  - Scala Logaritmica in y -> SetLogy
  - Griglia x-y->SetGridx, SetGridy
- Potevate usare anche l'Editor (ridondanza)



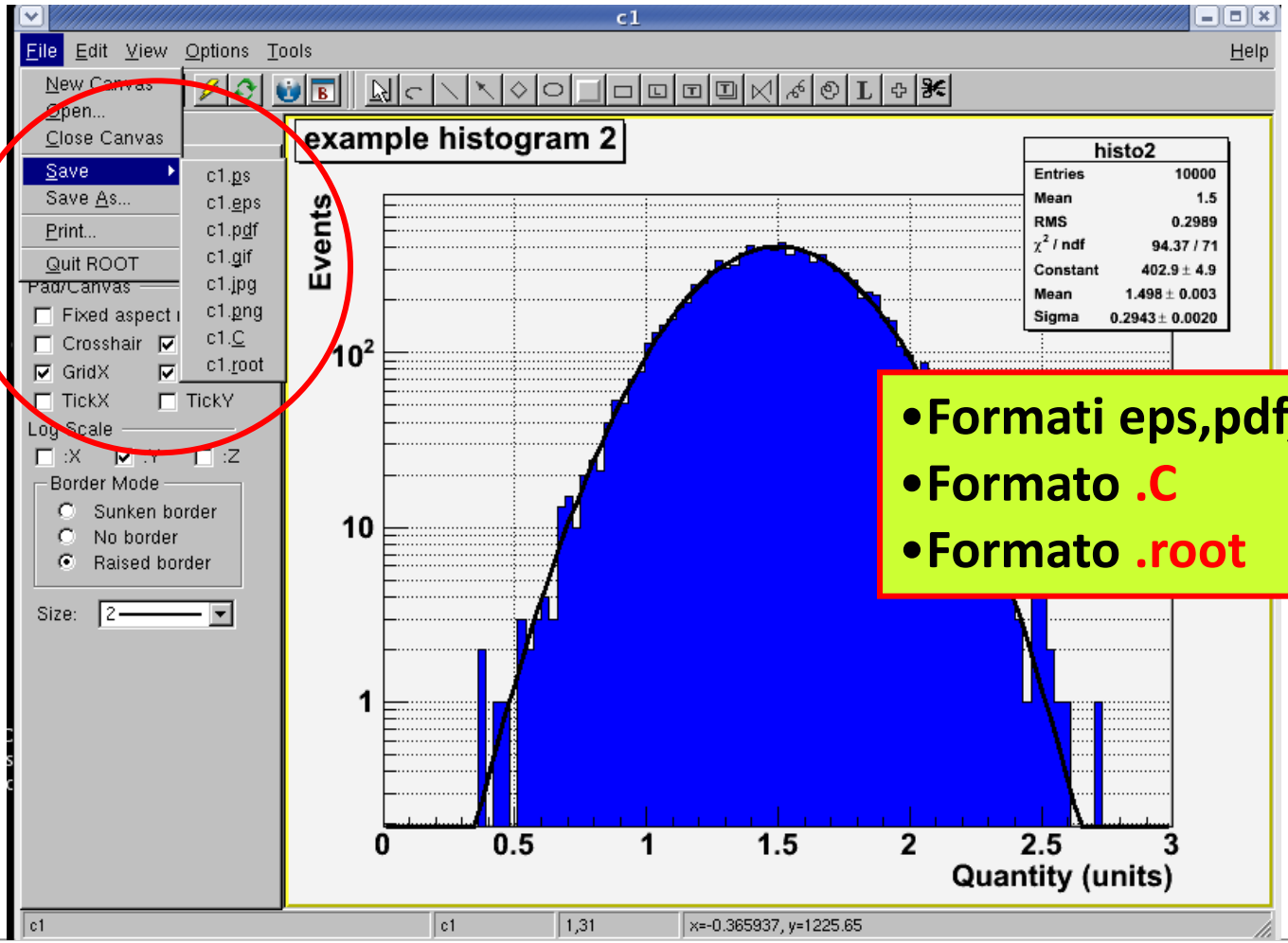
ROOT-GUI

- **Analogamente, per modificare i parametri degli assi del grafico potete sia usare il bottone dx+menu che l'Editor...**



ROOT-GUI

Per Salvare il vostro lavoro, diverse possibilità:
Nel menu File della canvas, selezionare Save



- **Formati eps,pdf,png,jpg,gif**
- **Formato .C**
- **Formato .root**

ROOT-GUI

emacs@pctof2.bo.infn.it

File Edit Options Buffers Tools C++ Help

```
{
//=========Macro generated from canvas: c1/c1
//=========  (Thu Apr 29 13:35:56 2010) by ROOT version5.25/04
   TCanvas *c1 = new TCanvas("c1", "c1",274,257,662,605);
   c1->ToggleEventStatus();
   c1->Range(-0.3716338,-1.172082,3.37702,3.354644);
   c1->SetBorderSize(2);
   c1->SetLogy();
   c1->SetGridx();
   c1->SetGridy();
   c1->SetFrameFillColor(0);

   TH1F *histo2 = new TH1F("histo2"
   histo2->SetBinContent(13,2);
   histo2->SetBinContent(15,1);
   histo2->SetBinContent(16,1);
   histo2->SetBinContent(18,3);
   histo2->SetBinContent(19,2);
   histo2->SetBinContent(20,3);
   histo2->SetBinContent(21,4);
   histo2->SetBinContent(22,3);
   histo2->SetBinContent(23,13);
   histo2->SetBinContent(24,15);
   histo2->SetBinContent(25,10);
   histo2->SetBinContent(26,20);
   histo2->SetBinContent(27,24);
   histo2->SetBinContent(28,21);
```

--:-- c1.C (C++ Abbrev)--L24--Top--

Beginning of buffer

- Il formato .C salva il grafico in forma di comandi di root in C++
- per riprodurre il grafico eseguite la macro:

root[] .x c1.C

ROOT-GUI

- Il formato .root salva la canvas e tutti gli oggetti di root contenuti
- per ricreare il grafico aprite il root file c1.root e fate doppio click sulla canvas
- Tutti gli oggetti sono disponibili per esser ulteriormente manipolati