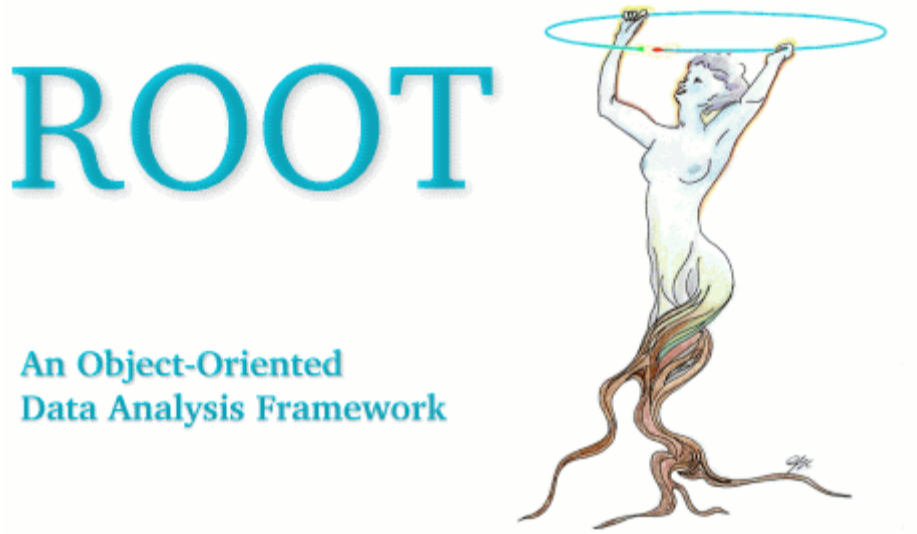
ROOT Tutorial



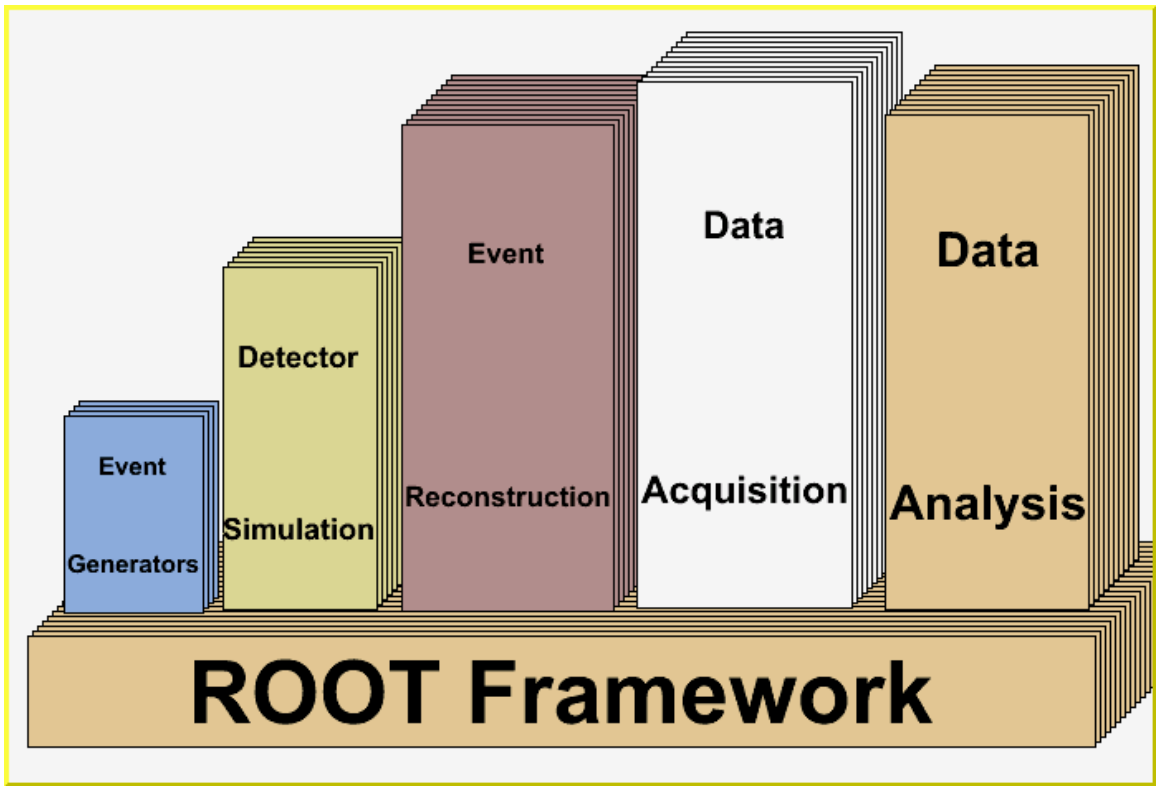
http://root.cern.ch

Perché ROOT?

Analisi dati e simulazione su larga scala

Il progetto ROOT nasce nell'ambito dell'esperimento NA49 al CERN
10 Terabytes di dati al giorno

ROOT framework:



L'architettura di ROOT

Si basa sul concetto di Classi Oggetti e Metodi…

Classi di…

✓Fisica
✓Vettori e Matrici
✓Istogrammi e Fit
✓Tree e nTuple
✓Grafica 2D
✓Grafica 3D (geometria dei detector)
✓Image processing
✓Neural Network
✓Graphical User Interface
✓C++ interpreter
✓Interfaccia col sistema operativo
✓Interfaccia SQL
✓Documentazione

Es: > cd %ROOTSYSTEM%tutorials
> root
root[0] .x demos.C

Richiami di C++ per fare pratica con Root…

http://www-root.fnal.gov/root/CPluPlus/index.html

Esempio 1: aprire e chiudere una sessione

```
>root
root[0] > .q
```

Esempio 2: un semplice calcolo

```
root[0] > int_t a;
root[1] > a = 3*5;
root[2] > cout << a << endl;
15
```

Esempio 3: eliminare il ";"

Esempio 4: utilizzo della classe TMath (set di funzioni matematiche)

```
root[0] > Double_t  x=1.3, y;
root[1] > TMath::Sin(TMath::Pi())
1.2246063538223730e-016
root[2] > y = TMath::Sin(TMath::Pi()) * x
1.5919882596909060e-016
```

Operatori aritmetici e di assegnazione

C++	Purpose	FORTRAN
x++	Postincrement	
++x	Preincrement	
x--	Postdecrement	
--x	Predecrement	
+x	Unary plus	+X
-x	Unary minus	-X
x*y	Multiply	X*Y
x/y	Divide	X/Y
x%y	Modulus	MOD(X,Y)
x+y	Add	X+Y
x-y	Subtract	X-Y
pow(x,y) or TMath::Power(x,y)	Exponation	X**Y (FORTRAN and CINT)
x = y	Assignment	X = Y
x += y (-=,*=,/=,%=,...)	Updating assignment	X = X+Y (X=X-Y,X=X*Y,...)

Esempio 5: un semplice script1

file mult.c:

```
Double_t mult(Double_t a, Double_t b) {
        return a*b;
}
```

```
root[0] > .L mult.c
root[1] >  mult(3,4)
(Double_t)1.20000000000000000e+001
```

Esempio 6: un semplice script2

file hello.c

```
void hello(char *name="") {
// print hello name
cout << "hello " << name << endl;
}
```

```
root [0] .L hello.C
root [1] hello()
hello
root [2] hello("Peter")
hello Peter
```

Esempio 7: un semplice script3

file multimult.c

```
Double_t mult(Double_t a, Double_t b = 2) {
// multiply a and b
return a*b;
}
Double_t mult() {
// prompt for two numbers and return the product
Double_t a, b;
cout << "enter a number:";       // output prompt string
cin >> a;                        // read input
cout << "enter a second number:"; // output prompt
cin >> b;                        // read input
return a*b;                      // return a*b
}
```

```
root [0] .L multimult.C
root [1] mult(3,4)
(Double_t)1.20000000000000000e+01
root [2] mult(3)
(Double_t)6.00000000000000000e+00
root [3] mult()
enter a number:4
enter a second number:8
(Double_t)3.20000000000000000e+01
```

Esempio 8: Valori logici e operatori di relazione

```
root [0] 3 < 4
(int)1
root [1] 4 < 3
(int)0
root [2] !(3<4)
(int)0
root [3] kTRUE
(Bool_t)1
root [4] kFALSE
(Bool_t)0
root [5] true
(enum bool)1
root [6] false
(enum bool)0
```

Valori e Operatori Logici, Operatori di Relazione

C++	ROOT extension	Purpose	FORTRAN
false or 0	kFALSE	False value	.FALSE.
true or nonzero	kTRUE	True value	.TRUE.
!x		Logical negation	.NOT.X
x && y		Logical and	X .AND. Y
x || y		Logical or	X .OR. Y
x < y		Less than	X. LT. Y
x <= y		Less than or equal	X. LE. Y
x > y		Greater than	X. GT. Y
x >= y		Greater than or equal	X .GE. Y
x == y		Equal	X. EQ. Y
x != y		Not equal	X. NE. Y

Esempio 9: Variabili (oggetti) e Tipi (Classi) in dettaglio…

Un programma è una procedura per alterare il valore di un oggetto
Un oggetto è un'area della memoria del computer che ha:

- ✓ un valore
- ✓ un tipo (una classe) che specifica come interpretare e manipolare l'oggetto

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

In C++ esiste la possibilità di definire i propri "data types" (CLASSI) con le operazioni ad essi associate (METODI).
In questo esempio usiamo la classe TDatime
(http://root.cern.ch/root/html//TDatime.html)

```
root [0] TDatime now;
root [1] now.Print();
Date/Time = Wed Oct 20 14:00:29 1999
root [2] TDatime yesterday;
root [3] yesterday.Print();
Date/Time = Wed Oct 20 14:01:15 1999
root [4] yesterday.Set(now.GetDate()-1,0);
root [5] yesterday.Print();
Date/Time = Tue Oct 19 00:00:00 1999
root [6] TDatime firstApril99;
root [7] firstApril99.Set(99,4,1,0,0,0);
root [8] firstApril99.Print();
Date/Time = Thu Apr  1 00:00:00 1999
```

La dichiarazione di un oggetto chiama il suo COSTRUTTORE cioè un metodo con lo stesso nome della classe

Esempio 10: uso dei puntatori

```
root [0] TDatime now;
root [1] TDatime *p;
root [2] p = &now;
root [3] p->Print();
Date/Time = Wed Oct 20 17:13:42 1999
root [4] (*p).Print();
Date/Time = Wed Oct 20 17:13:42 1999
```

Nell'istruzione [1] viene dichiarato un puntatore all'oggetto TDatime.
In C++ si usa:

```
oggetto.metodo()
punatoreaoggetto→metodo()
```

Esempio 11: uso del "new"

```
root [0] TDatime *now = new TDatime();
root [1] now->Print();
Date/Time = Fri Oct 22 09:43:39 1999
```

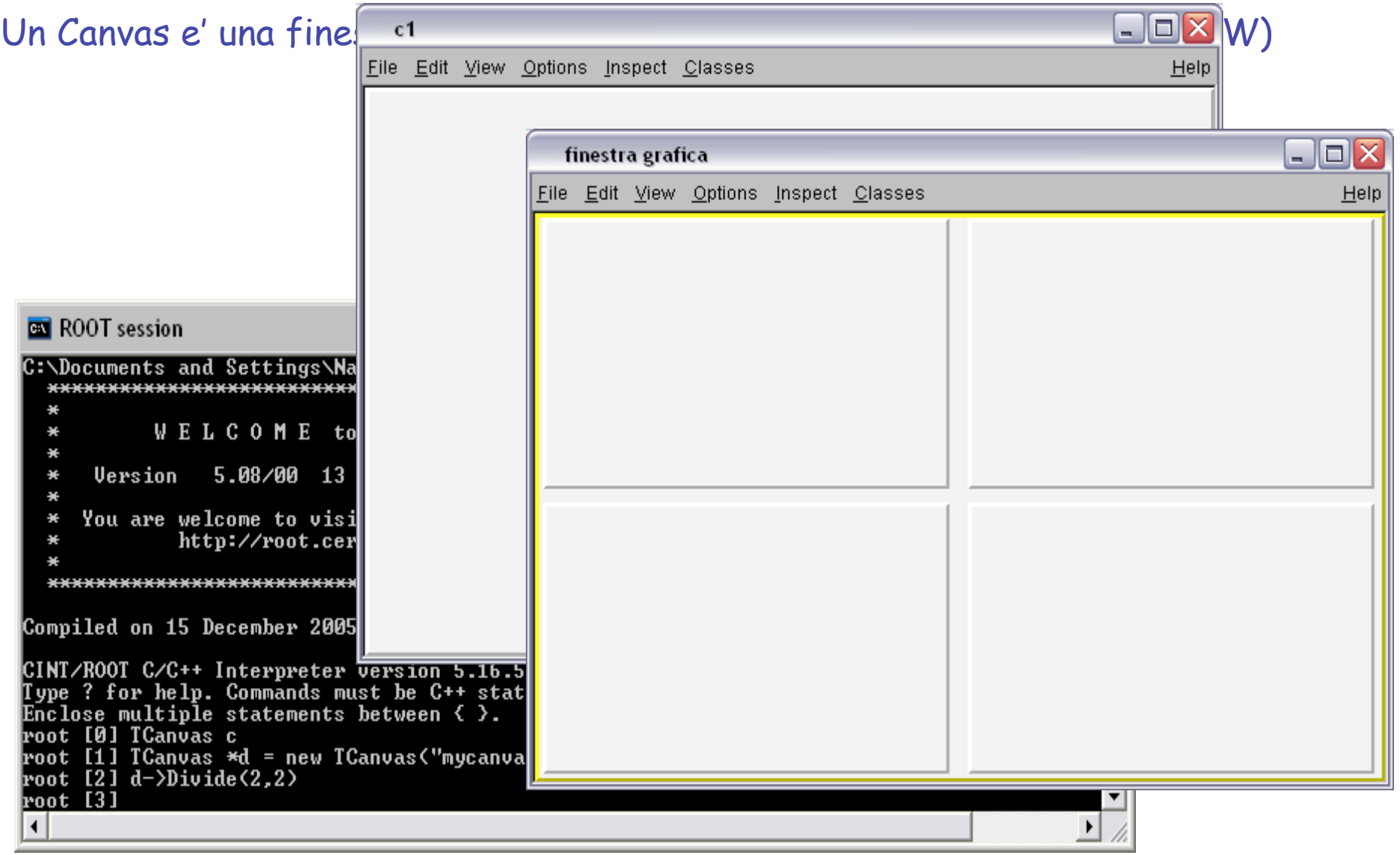
Nell'istruzione [0] viene dichiarto un puntatore di nome "now" all'oggetto TDatime e viene inizializzato con l'indirizzo di un nuovo oggetto TDatime.

```
new   costruttore(parametri)
```

Analisi Dati di base con ROOT

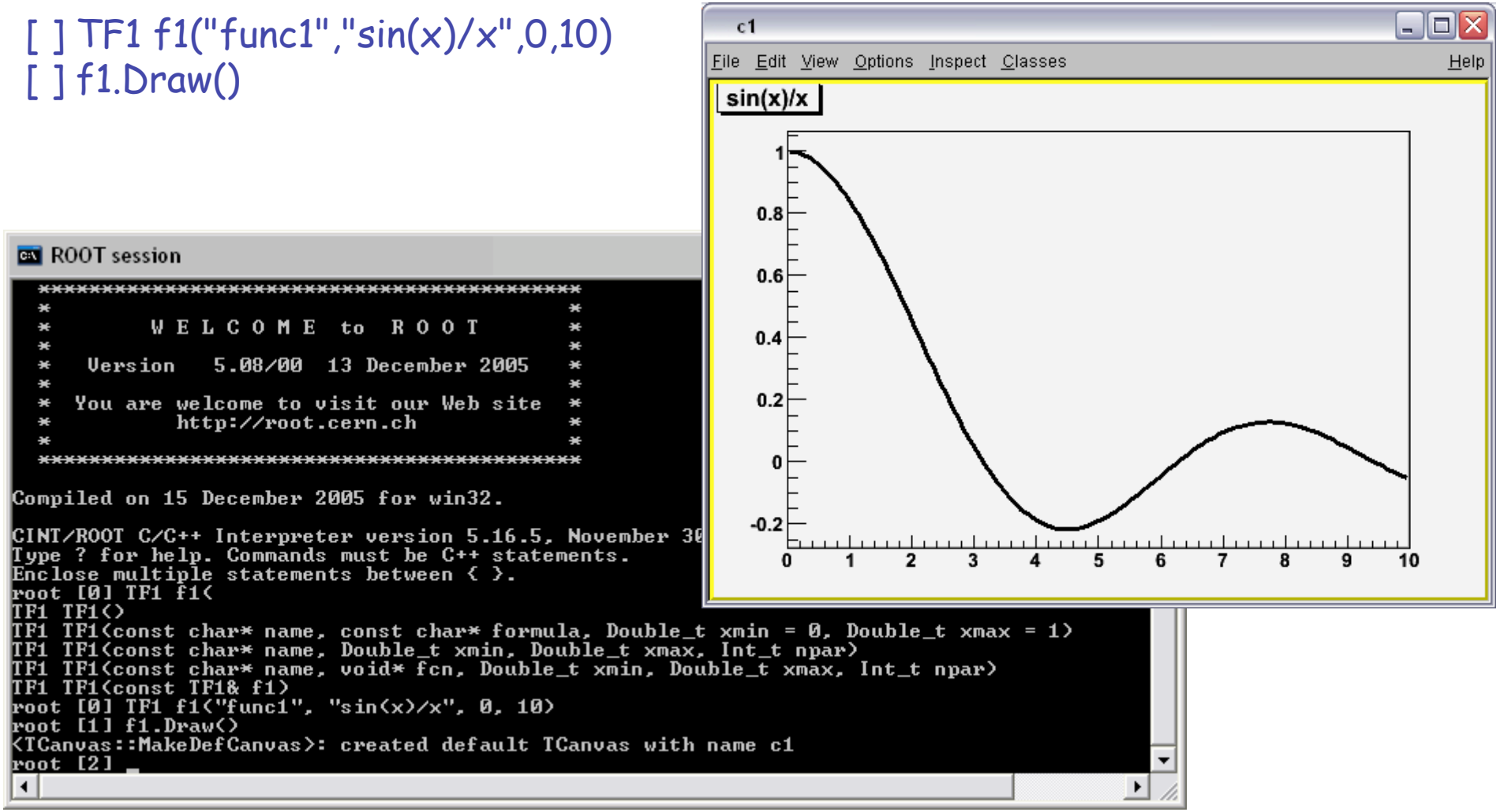
CANVAS

Un Canvas e' una fine[illegible]W)



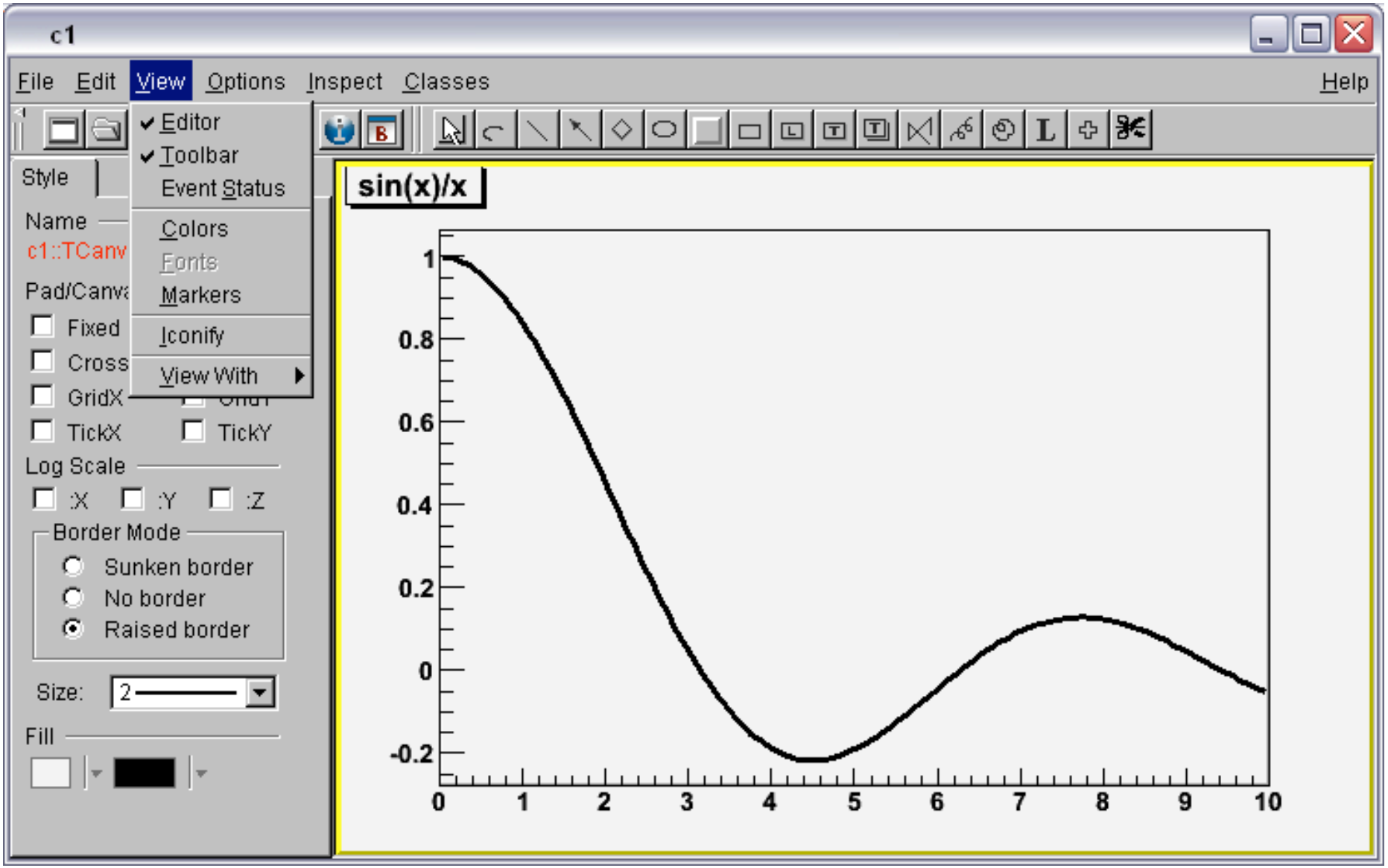
Plottare una funzione

```
[ ] TF1 f1("func1","sin(x)/x",0,10)
[ ] f1.Draw()
```



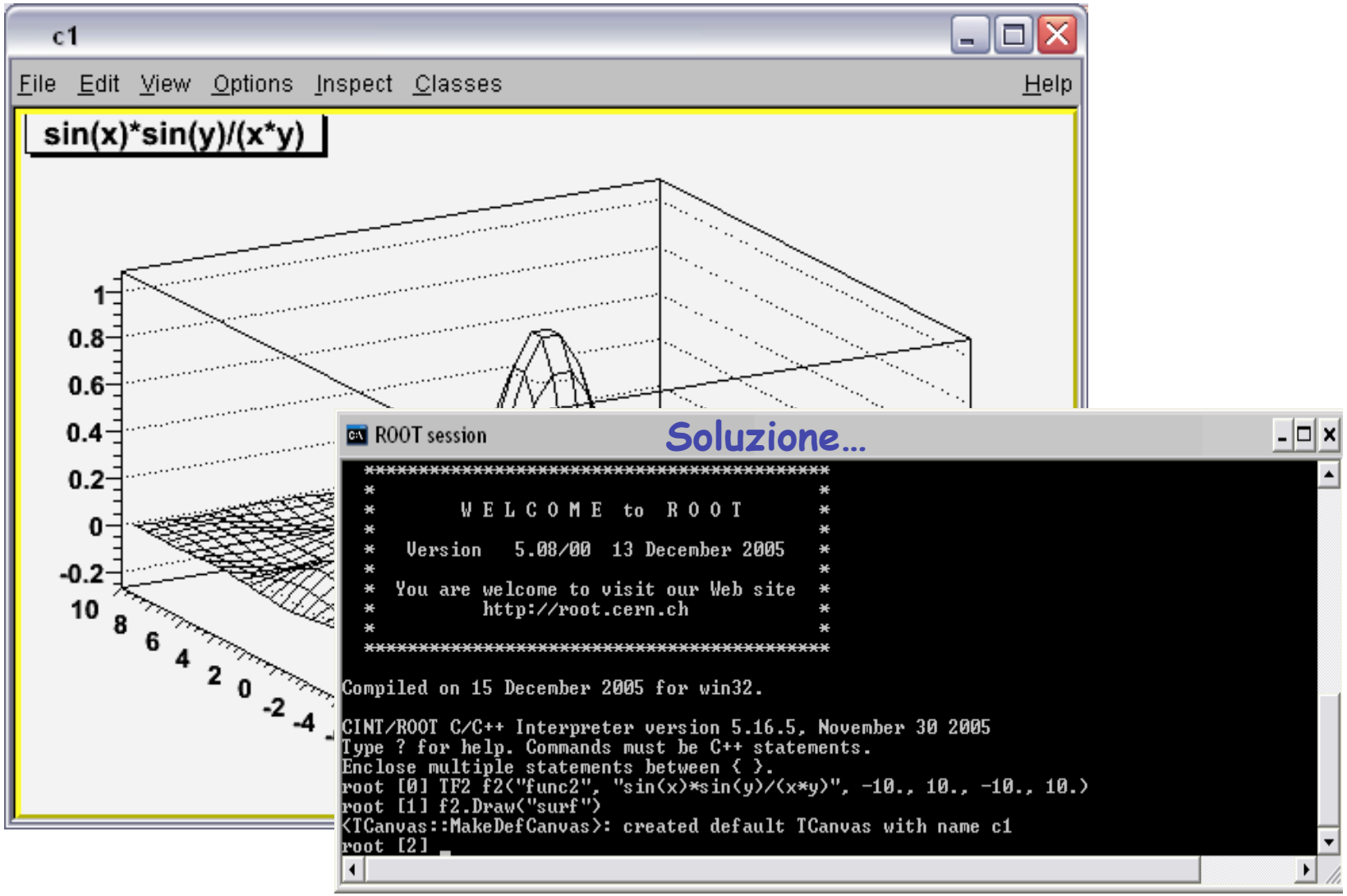
http://root.cern.ch/ ⇨
Reference Guide ⇨ The ROOT Class Categories ⇨ Histogram ⇨ TF1

GUI Basic


c1
File
Edit
View
Options
Inspect
Classes
Help
Editor
Toolbar
Event Status
Colors
Fonts
Markers
Iconify
View With
Style
Name
c1::TCanv
Pad/Canva
Fixed
Cross
GridX
TickX
TickY
Log Scale
:X
:Y
:Z
Border Mode
Sunken border
No border
Raised border
Size:
2
Fill
sin(x)/x
1
0.8
0.6
0.4
0.2
0
-0.2
0
1
2
3
4
5
6
7
8
9
10

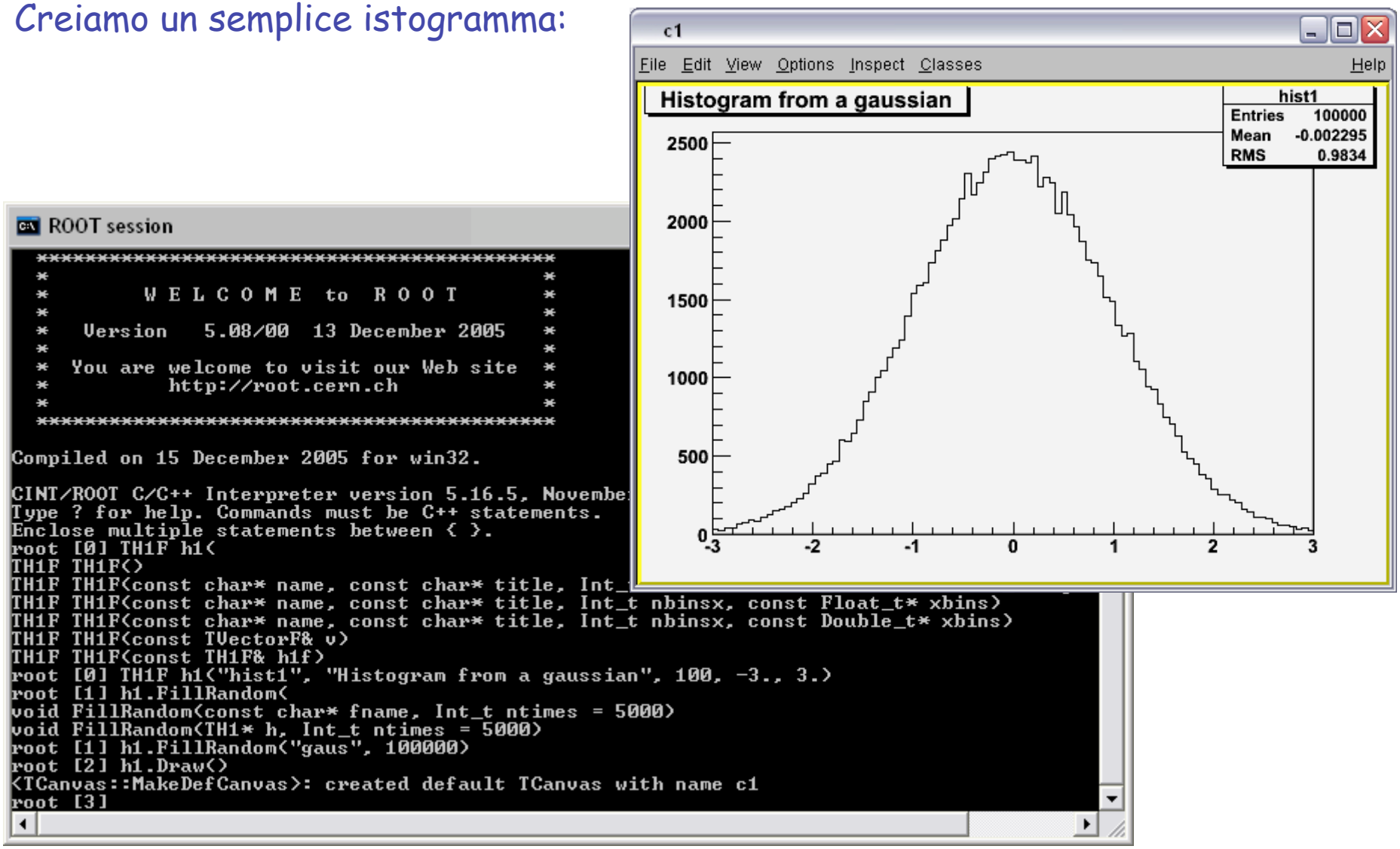
Esercizio 1:

Riprodurre il plot seguente:



Istogrammi

Creiamo un semplice istogramma:



Esercizio 2:

Riempire un istogramma estraendo 100000 numeri random dalla funzione:

$$par(0)e^{-0.5\left(\frac{x-par(1)}{par(2)}\right)^2}$$

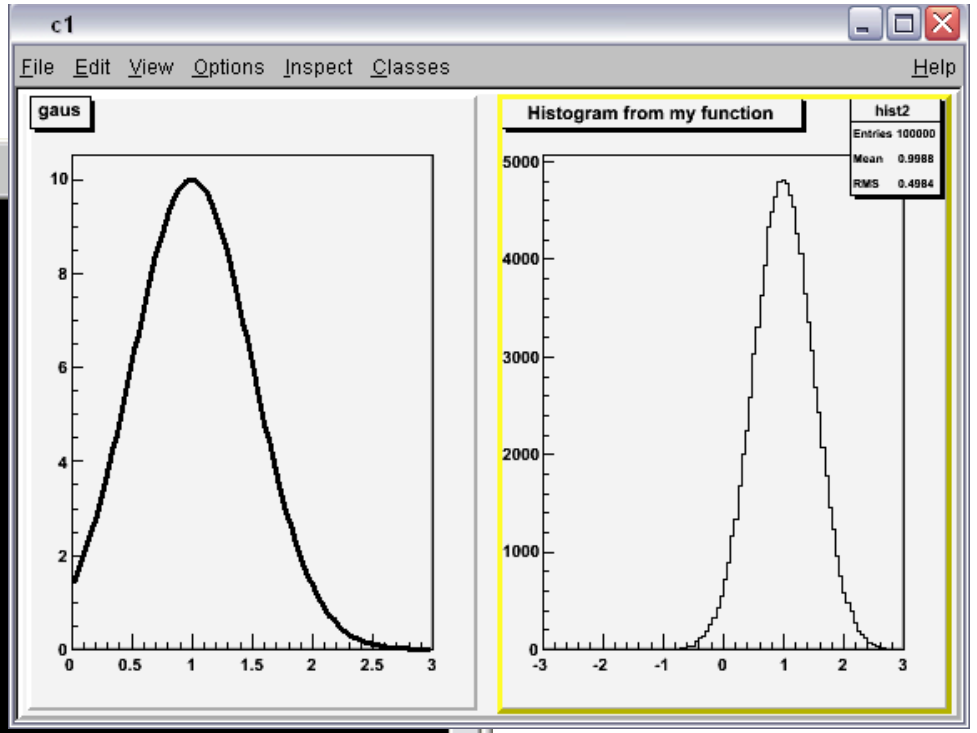
con par(0)=10, par(1)= 1, par(2)= 0.5

Soluzione…

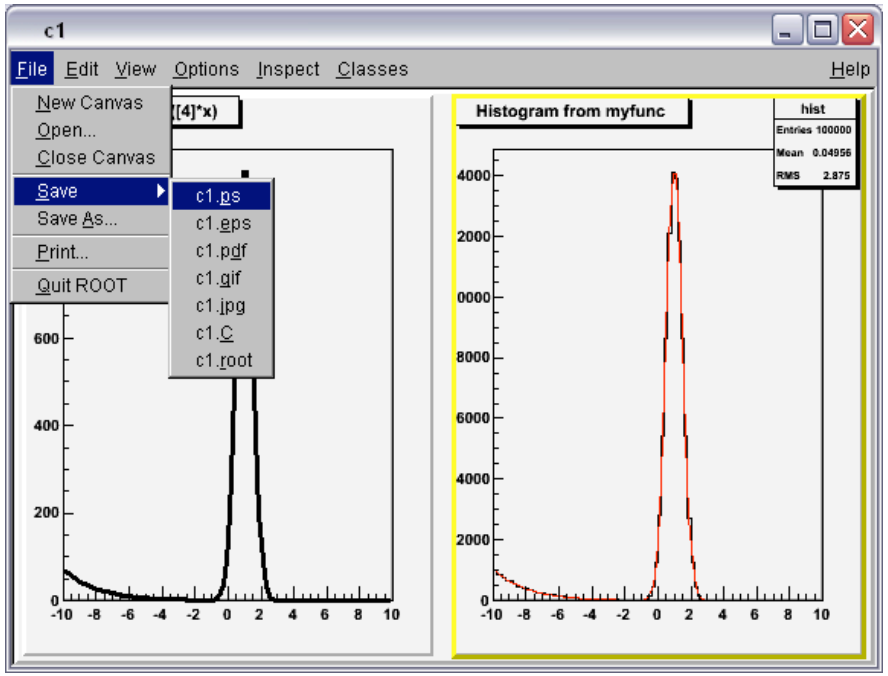
```
ROOT session
  ********************************************
  *                                          *
  *        W E L C O M E  to  R O O T        *
  *                                          *
  *   Version   5.08/00  13 December 2005    *
  *                                          *
  *  You are welcome to visit our Web site   *
  *          http://root.cern.ch             *
  *                                          *
  ********************************************

Compiled on 15 December 2005 for win32.

CINT/ROOT C/C++ Interpreter version 5.16.5, November 30 2005
Type ? for help. Commands must be C++ statements.
Enclose multiple statements between { }.
root [0]
root [0] TF1 myfunc("myfunc", "gaus", 0, 3)
root [1] myfunc.SetParameters(10., 1., 0.5)
root [2] TCanvas c
root [3] c->Divide(2,1)
root [4] c->cd(1)
(class TVirtualPad*)0x7bd7a58
root [5] myfunc.Draw()
root [6] TH1F h2("hist2", "Histogram from my function", 100,-3, 3)
root [7] c->cd(2)
(class TVirtualPad*)0x7bd7d20
root [8] h2.FillRandom("myfunc", 100000)
root [9] h2.Draw()
root [10]
```



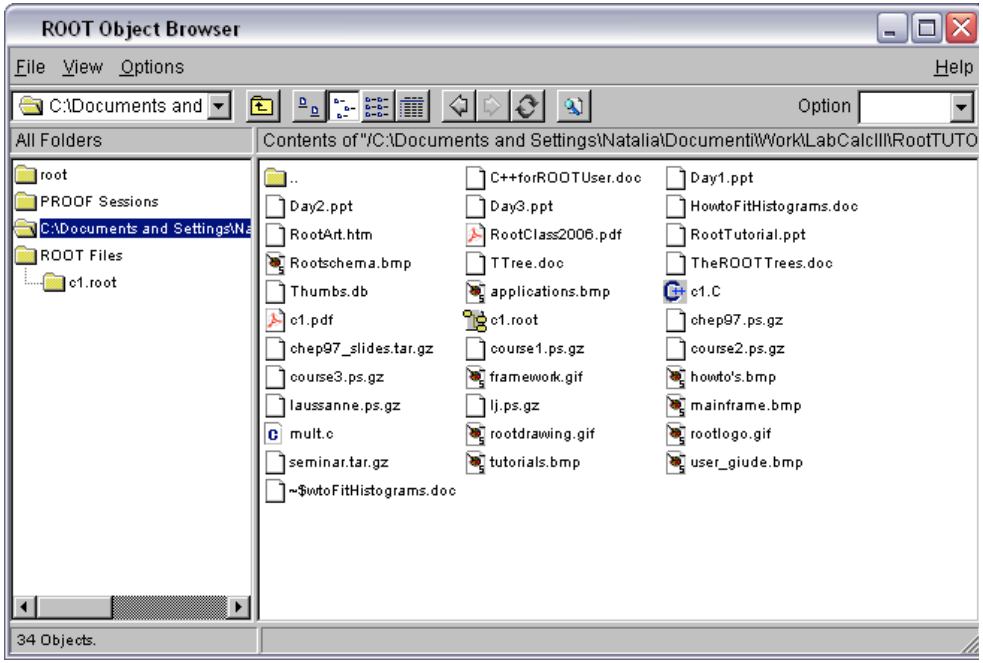
Salvare il proprio lavoro. Uso di TBrowser



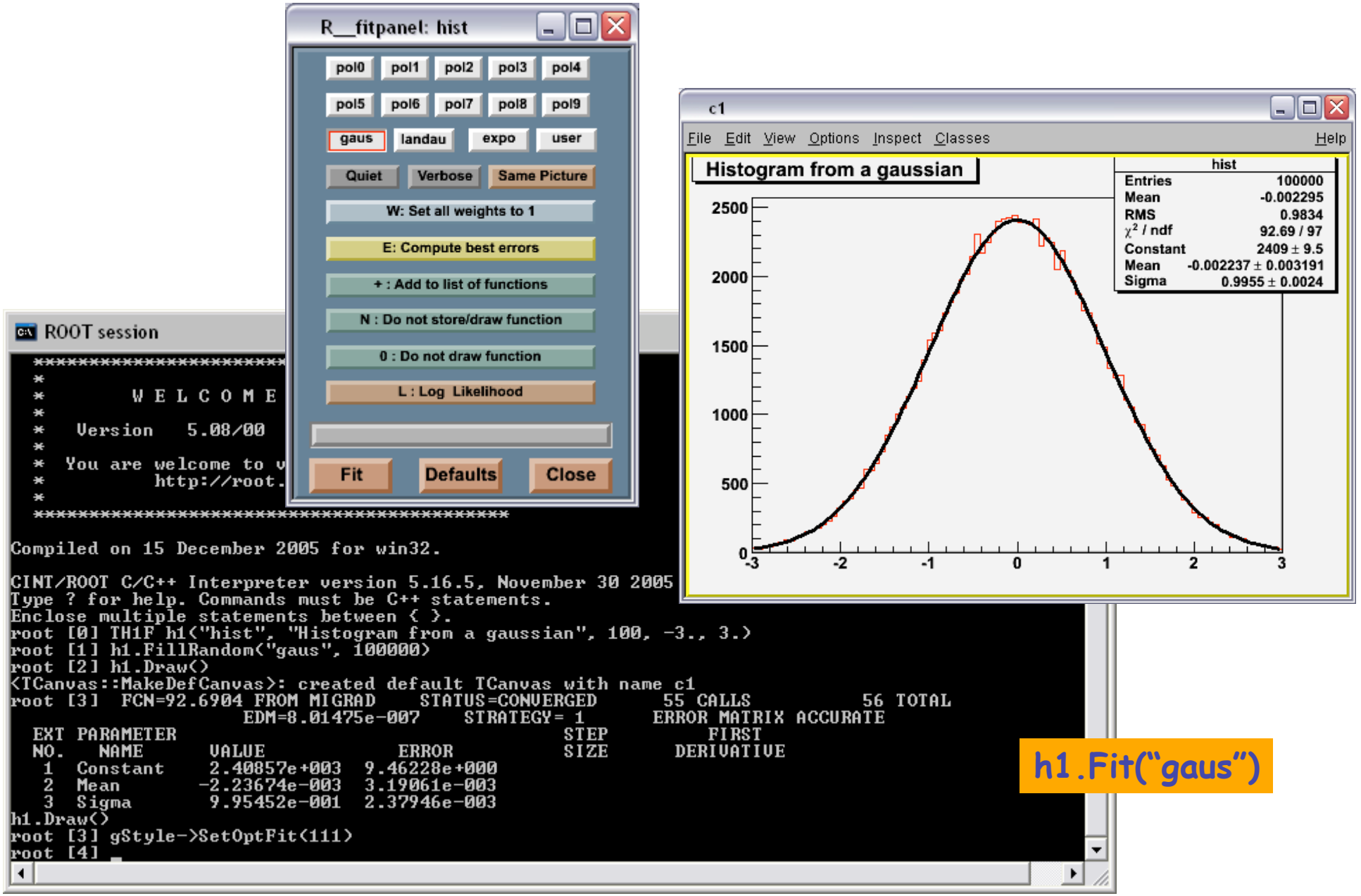
1) c1.C
Per riutilizzare il plot:
root[0]> .x c1.C

2) c1.pdf

3) c1.root
Per riutilizzare il plot:
root[0]> TBrowser tb



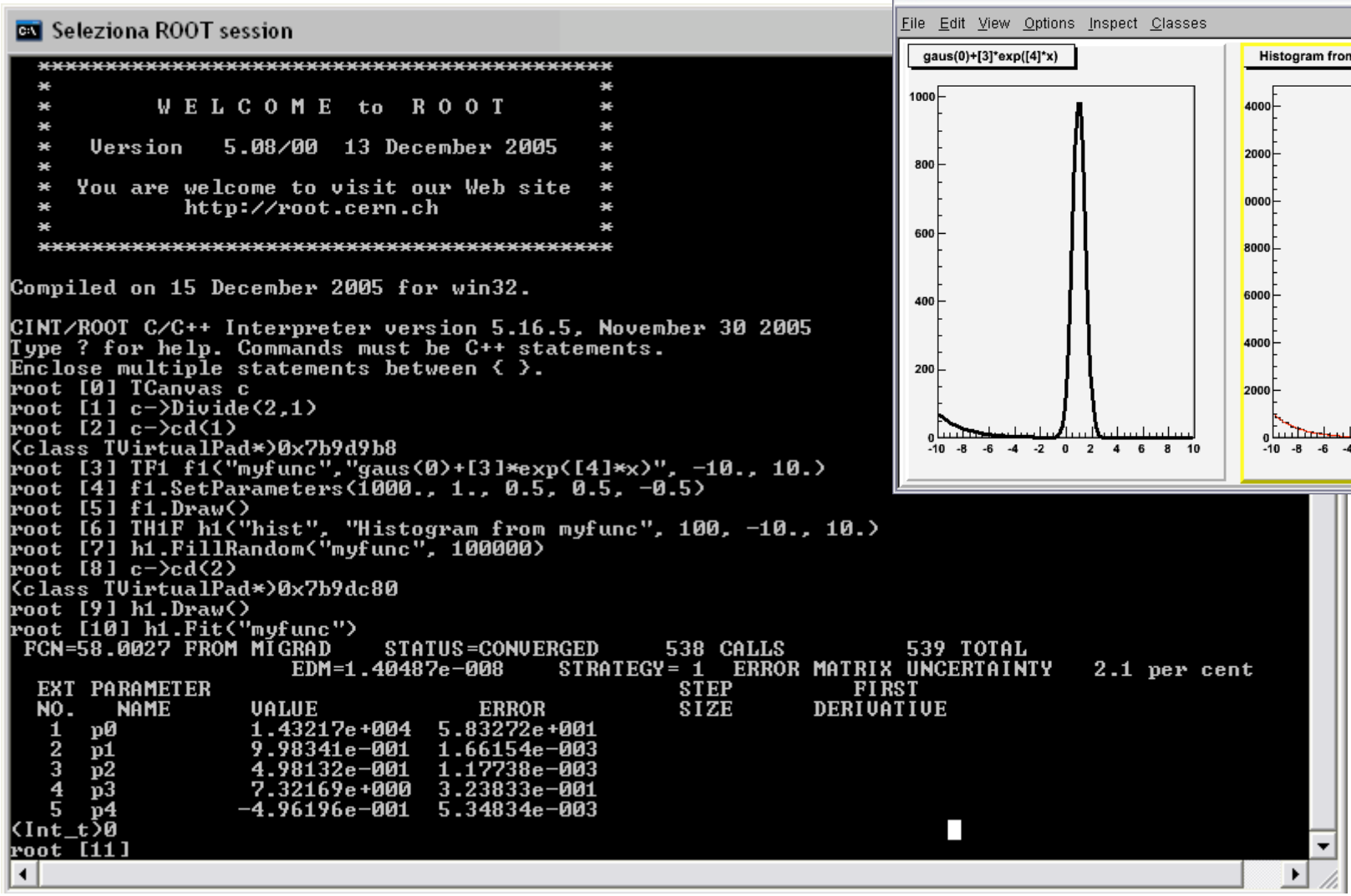
Fittare un istogramma 1



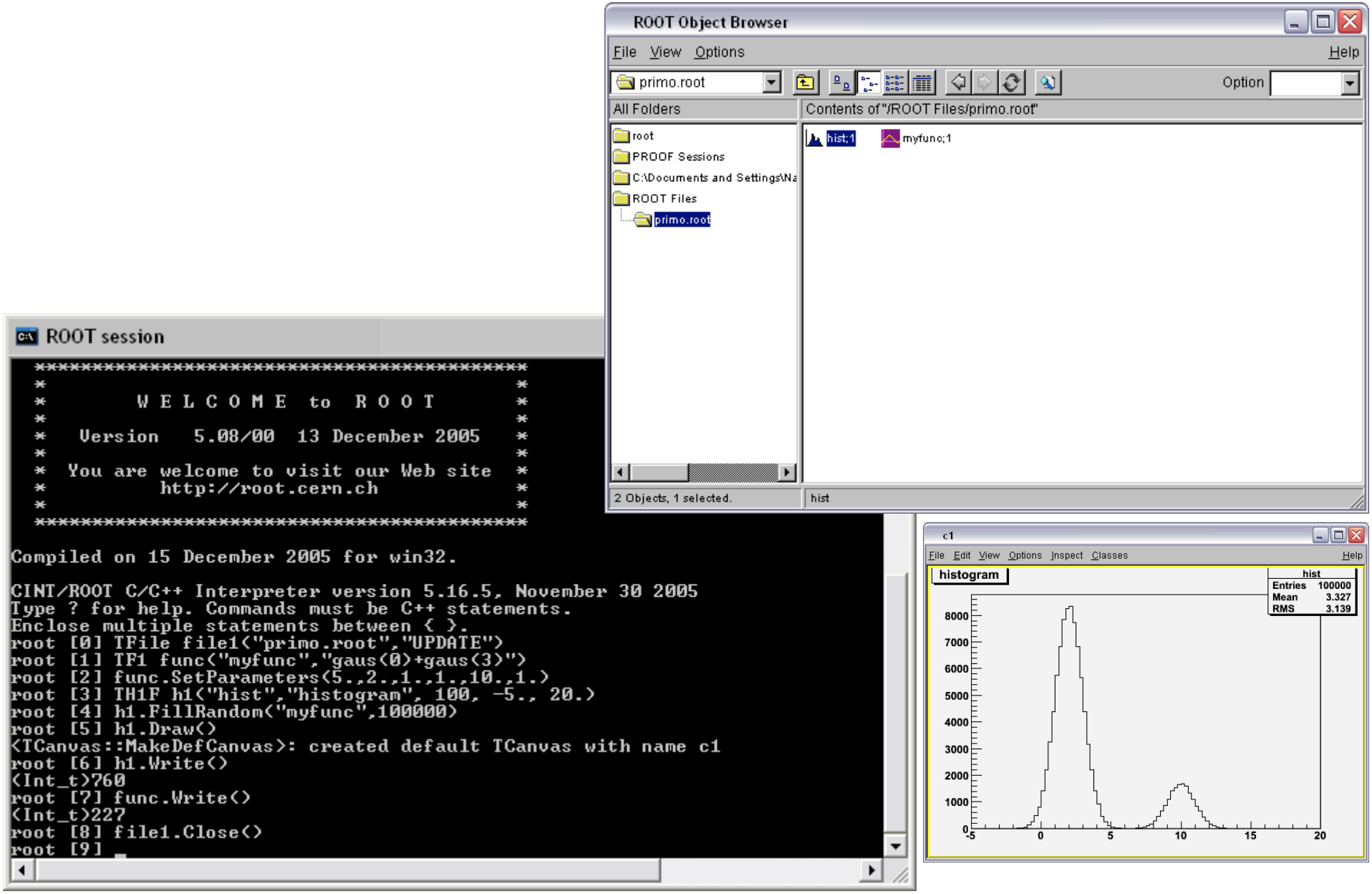
Fittare un istogramma 2

Esercizio 3:

fittare un istogramma con la funzione: $par(0)e^{-\left(\frac{x-par(1)}{par(2)}\right)^2} + par(3)e^{par(4)x}$



Root file


ROOT Object Browser
File View Options Help
primo.root
Option
All Folders
Contents of "/ROOT Files/primo.root"
root
PROOF Sessions
C:\Documents and Settings\Na
ROOT Files
primo.root
hist;1
myfunc;1
2 Objects, 1 selected.
hist
ROOT session
W E L C O M E to R O O T
Version 5.08/00 13 December 2005
You are welcome to visit our Web site
http://root.cern.ch
Compiled on 15 December 2005 for win32.
CINT/ROOT C/C++ Interpreter version 5.16.5, November 30 2005
Type ? for help. Commands must be C++ statements.
Enclose multiple statements between { }.
root [0] TFile file1("primo.root","UPDATE")
root [1] TF1 func("myfunc","gaus(0)+gaus(3)")
root [2] func.SetParameters(5.,2.,1.,1.,10.,1.)
root [3] TH1F h1("hist","histogram", 100, -5., 20.)
root [4] h1.FillRandom("myfunc",100000)
root [5] h1.Draw()
(TCanvas::MakeDefCanvas): created default TCanvas with name c1
root [6] h1.Write()
(Int_t)760
root [7] func.Write()
(Int_t)227
root [8] file1.Close()
root [9]
c1
File Edit View Options Inspect Classes Help
histogram
hist
Entries 100000
Mean 3.327
RMS 3.139
8000
7000
6000
5000
4000
3000
2000
1000
0
-5
0
5
10
15
20

Tree

Il TREE di ROOT è un'estensione del concetto di Ntupla di PAW

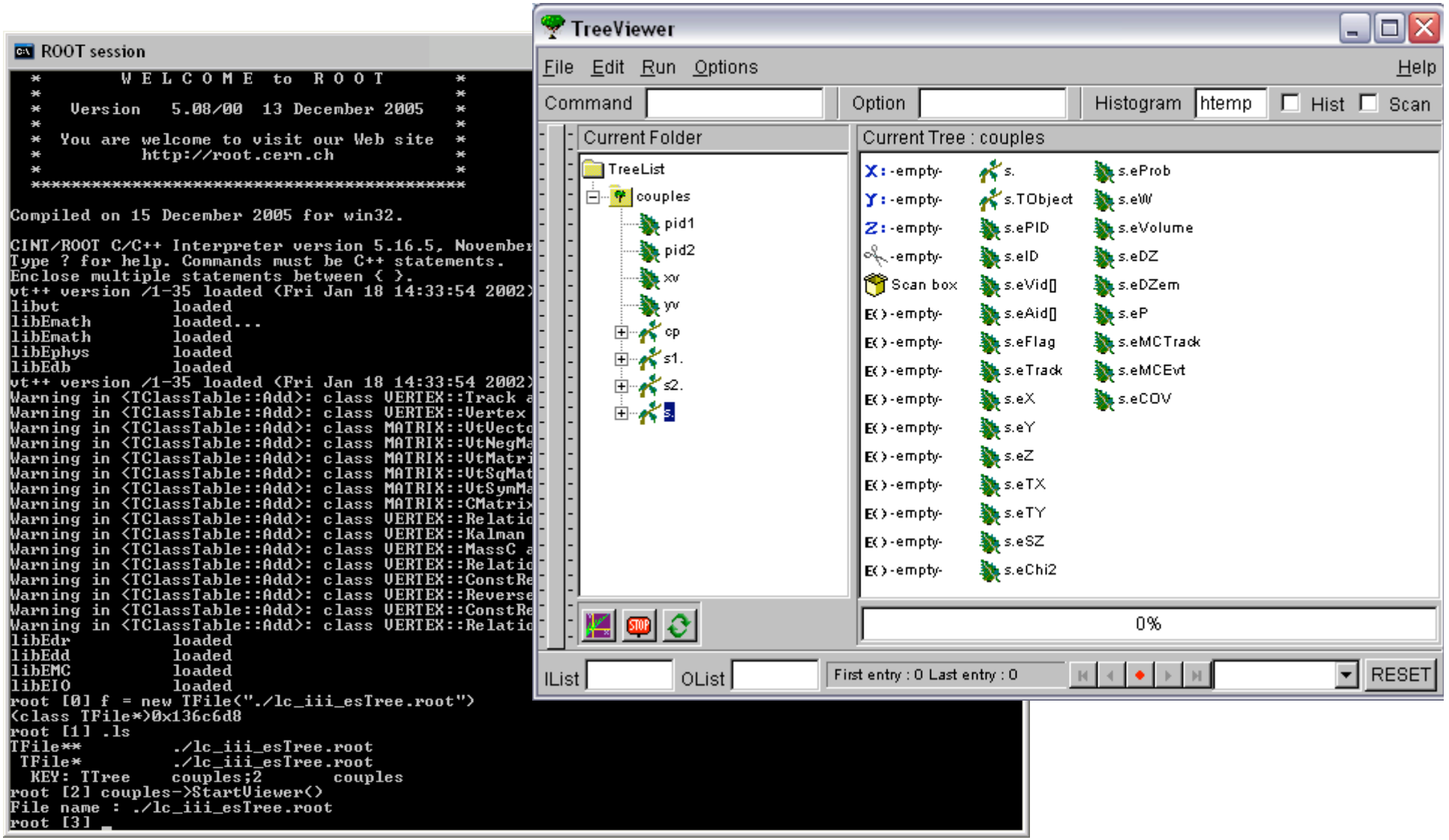
Le Ntuples possono contenere solo oggetti molto semplici (singole variabili o vettori)

Un **Tree** (classe TTree) è composto di Branches. Ogni branch (classe TBranch) è descritto dalle leaves (classe TLeaf). Le leaves possono essere semplici variabili, strutture, arrays o oggetti. Un array può essere di lunghezza variabile, e la lunghezza stessa può diventare una variabile nello stesso o in un altro branch. I Branches sono generalmente degli oggetti.

La struttura dei dati di un **Tree** permette di accedere direttamente ad ogni evento, ogni branch e ogni leaf.

Tree

Esempio: analizziamo il Tree contenuto nel file "lc_iii_esTree.root"



Esempio1: alcuni metodi utili per l'analisi di un Tree

```
root[ ]> treename->Scan()
root[ ]> treename->Print()
root[ ]> treename->Draw("varname")                                  1D
root[ ]> treename->Draw("varname1:varname2")                        2D
root[ ]> treename->Draw("varname1:varname2", "", "lego2")           2D con opzioni
root[ ]> treename->Draw("varname1:varname2", "varname>3", "lego")   2D con taglio
root[ ]> treename->Draw("varname1:varname2:varname3")               3D
root[ ]> treename->Fit("func", "varname")                           Fit
root[ ]> treename->Fit("func", "varname", "cut")                    Fit con taglio
```

Esempio2: uso della classe TCut per applicare un taglio

```
root[ ]> TCut cut1="varname1>0.3"
root[ ]> treename->Draw("varname1:varname2",cut1, "lego2")
root[ ]> TCut cut2="varname2<0.3*varname1+89"
root[ ]> treename->Draw("varname1:varname2",cut1 && cut2, "lego2")
```

Esempio3: uso degli Alias

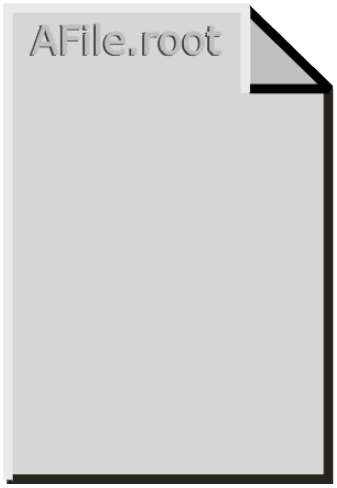
```
root[ ]> treename->SetAlias("aliasname","(var1*var2)/(sin(var3)*5667)")
root[ ]> treename->Draw("aliasname")
root[ ]> treename->Draw("varname4","aliasname!=908")
```

Per fare un Tree...

I 5 step per costruire un Tree:

1. Creare un TFile
2. Creare un TTree
3. Aggiungere TBranch al TTree
4. Riempire il tree
5. Scrivere il file

Step 1: Creare un oggetto TFile

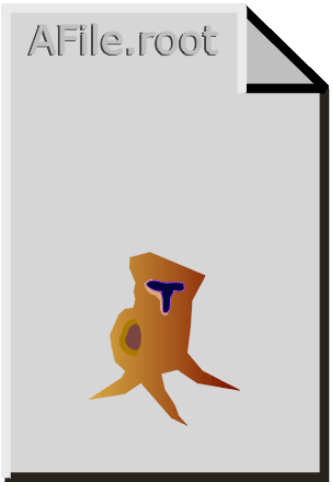


Il costruttore TFile

- ✓ file name (i.e. " es_FullMC.root ")
- ✓ option: NEW, CREATE, RECREATE, UPDATE, or READ

```
TFile *myfile = new TFile("es_FullMC.root","RECREATE");
```

Step 2: Creare un oggetto TTree



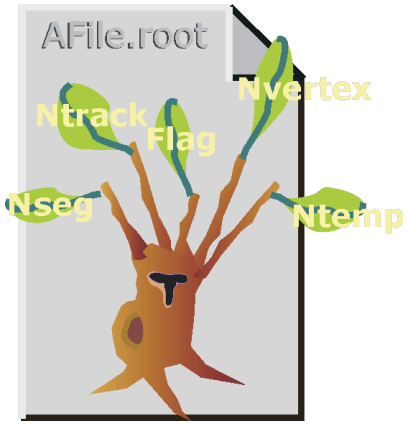
Il costruttore TTree:

- ✓ Tree Name (e.g. "myTree")
- ✓ Tree Title

```
TTree *tree = new TTree("myTree","A ROOT tree");
```

Step 3: Aggiungere i Branches

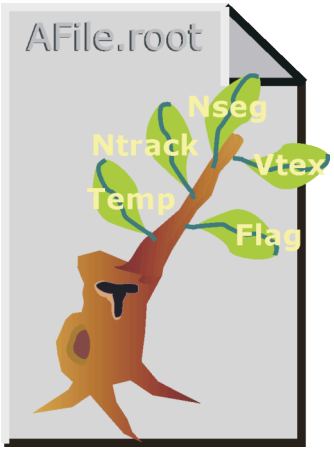
- ✓ Branch name
- ✓ Address of the pointer to the Object (descendant of TObject)
- ✓ LeafList




```
Event *event = new  Event();
myTree->Branch("EventBranch","Event",&event);
```

Step 3bis: Aggiungere i Branches con una lista di variabili

- ✓ Branch name
- ✓ Address: the address of the first item of a structure.
- ✓ Leaflist: all variable names and types
- ✓ Order the variables according to their size

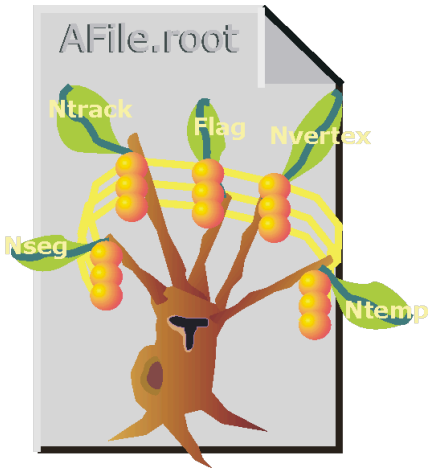



```
TBranch *b = tree->Branch("Ev_Branch",&event,"ntrack/I:nseg:nvtex:flag/i:temp/F");
```

Step 4: Riempire il Tree

- ✓ Create a for loop
- ✓ Assign values to the event object
- ✓ Call the Fill method for the tree

```
myTree->Fill()
```

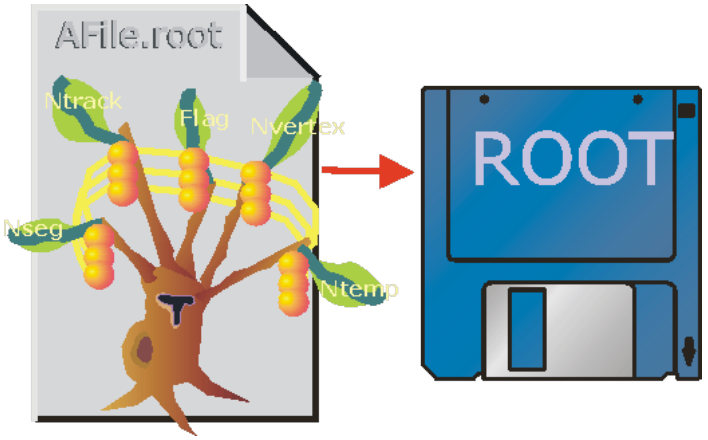


Step 5: Scrivere nel File

The TFile::Write()

- ✓ Writes Histograms and Trees
- ✓ Write is needed to write file header

```
hfile->Write();
```

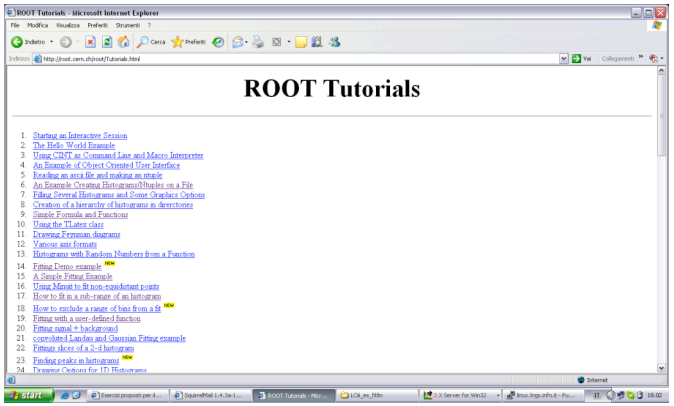


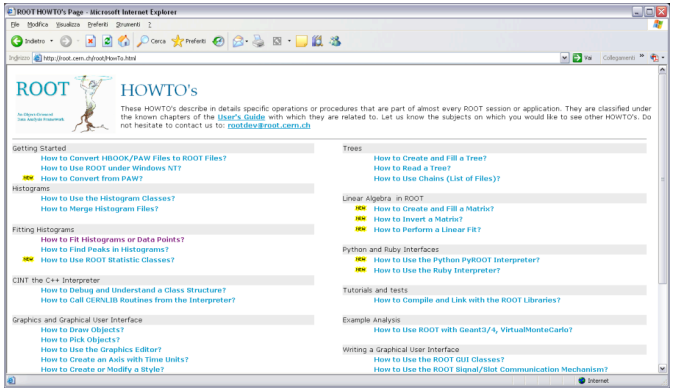
Esercizio 4:

creare il Tree dell'esercizio 9bis e analizzarlo
(http://www.aquila.infn.it/lc_iii/esercizi.html)

Esempi, suggerimenti, documentazione…

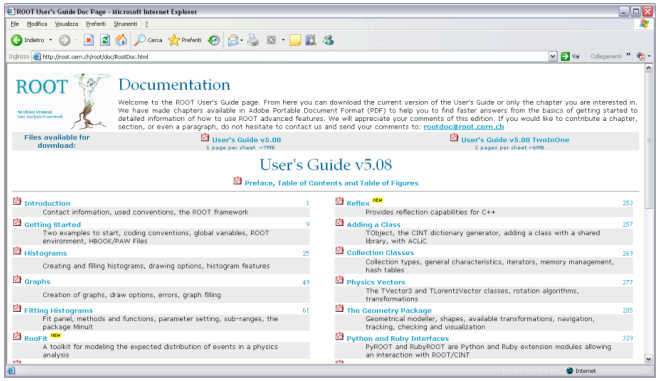
- ✓ ROOT Tutorials:
  http://root.cern.ch/root/Tutorials.html

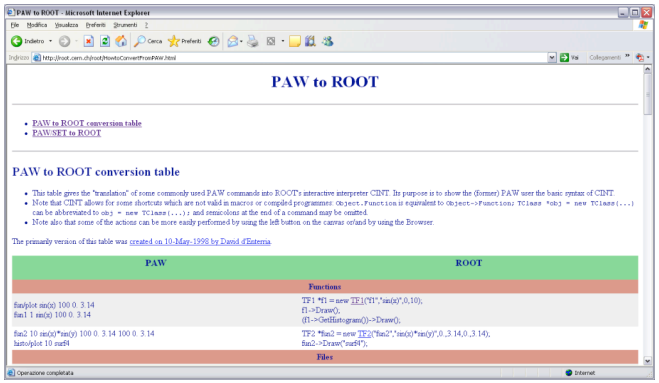




- ✓ ROOT How To:
  http://root.cern.ch/root/Howto.html

- ✓ Root User's Guide:
  http://root.cern.ch/root/doc/RootDoc.html





- ✓ PAW2ROOT:
  http://root.cern.ch/root/HowtoConvertFromPAW.html