EP578 Computing for Physicists

Topic 8

ROOT: Basics

Department of
Engineering Physics

University of Gaziantep

Course web page
www.gantep.edu.tr/~bingul/ep578

Oct 2011

In this lecture we will learn some fundamentals the ROOT program.

Users Guide and Reference Manuals are available at

http://root.cern.ch

ROOT

- ROOT
  is an “object oriented framework for data analysis”
  – read data from some source
  – write data (persistent objects)
  – selected data with some criteria
  – produce results as plots, numbers, fits, ...
- Supports “interactive” C/C++ (like Python) and “compiled” C++ usage
- Integrates several tools like random number generations, fit methods (Minuit), Neural Network framework
- Developed and supported by HEP community

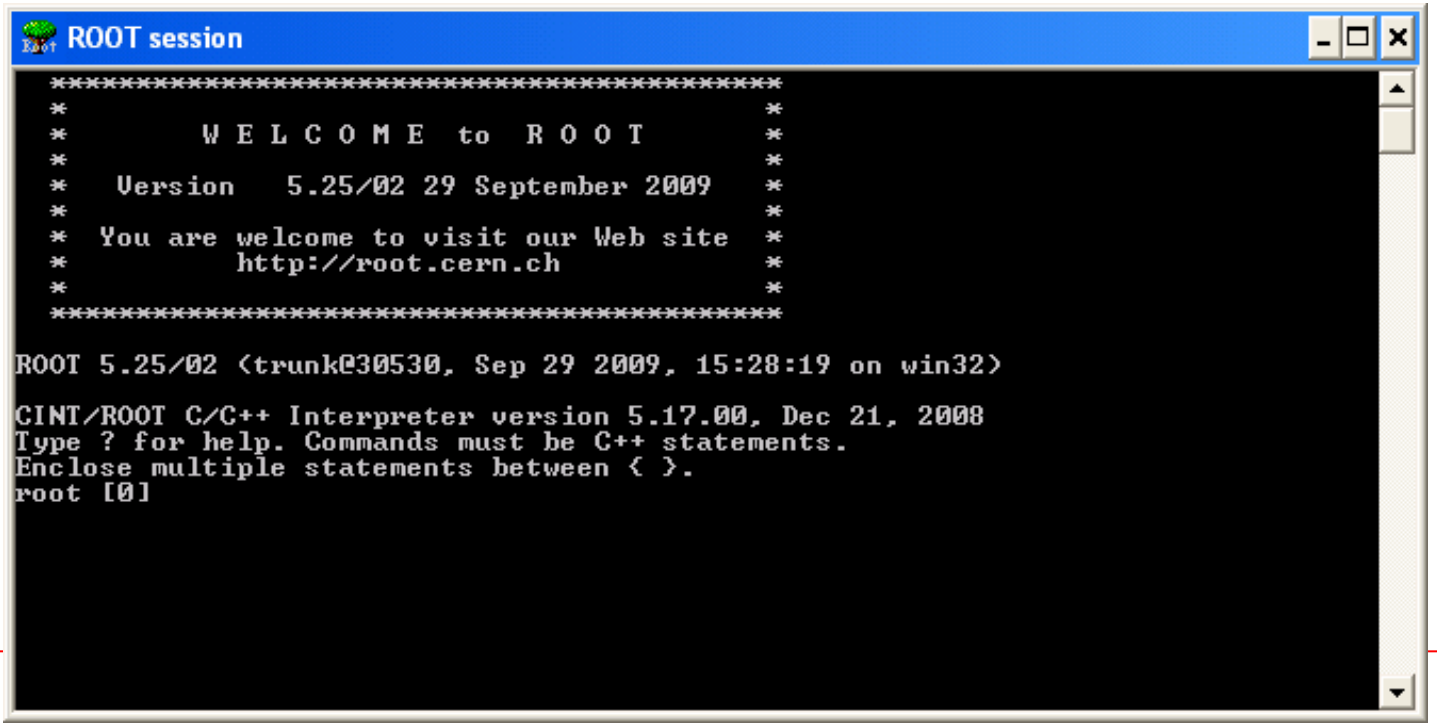
ROOT Console

- Launch ROOT interactive console (CINT interpreter)

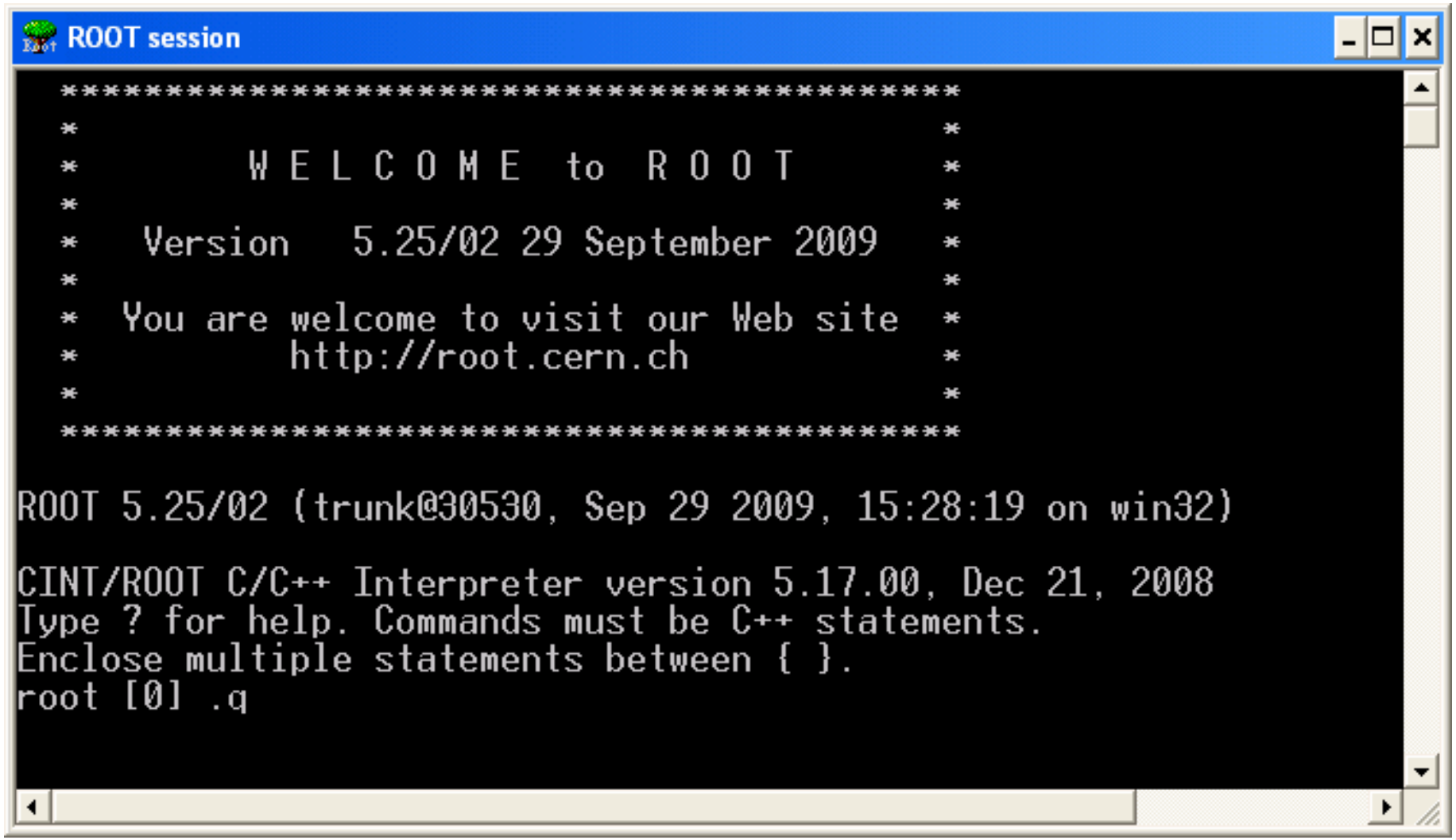
```
> root
```

or

```
> root.exe
```



- To quit from ROOT type: **.q**

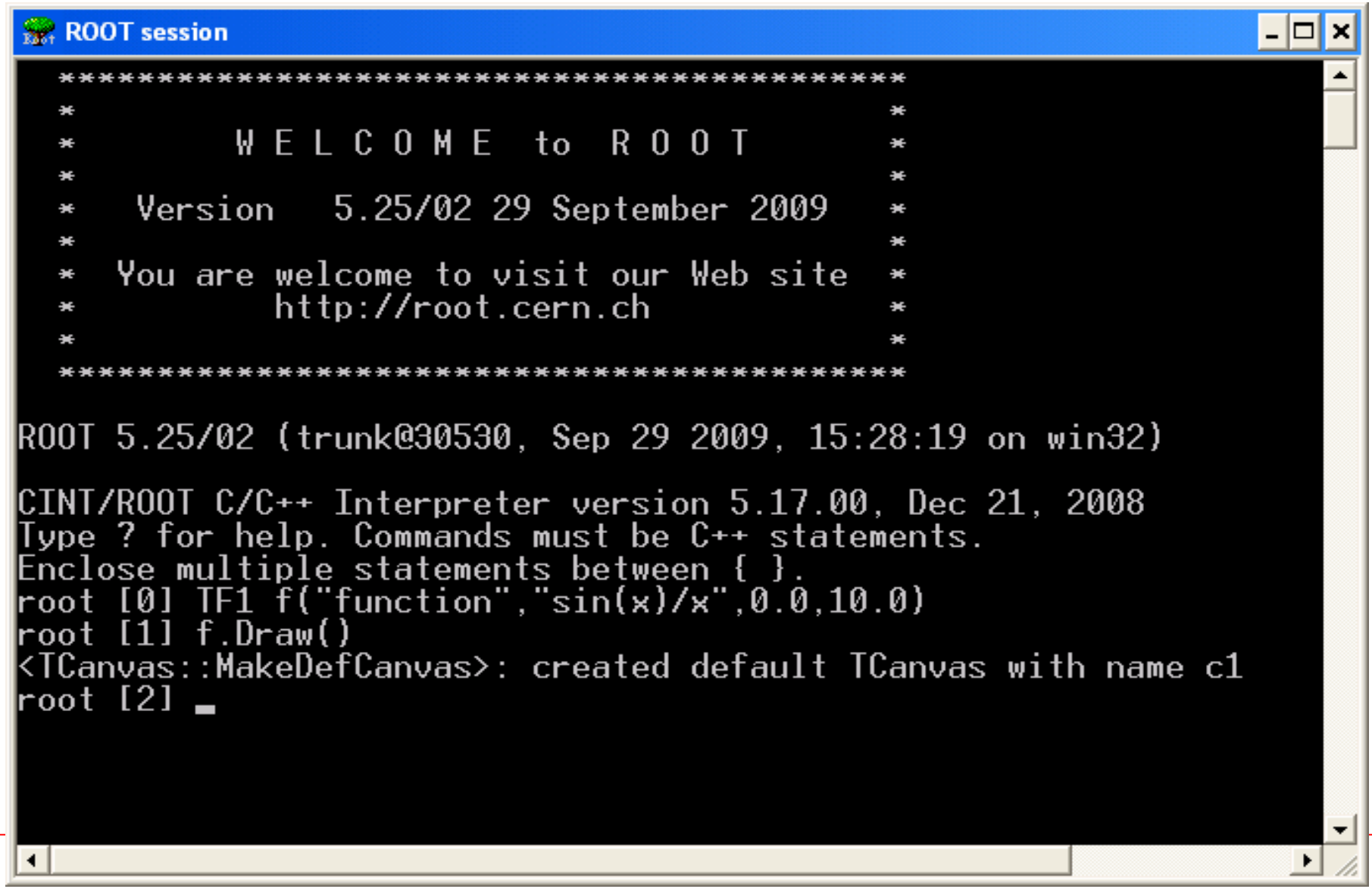


Basic Stuff

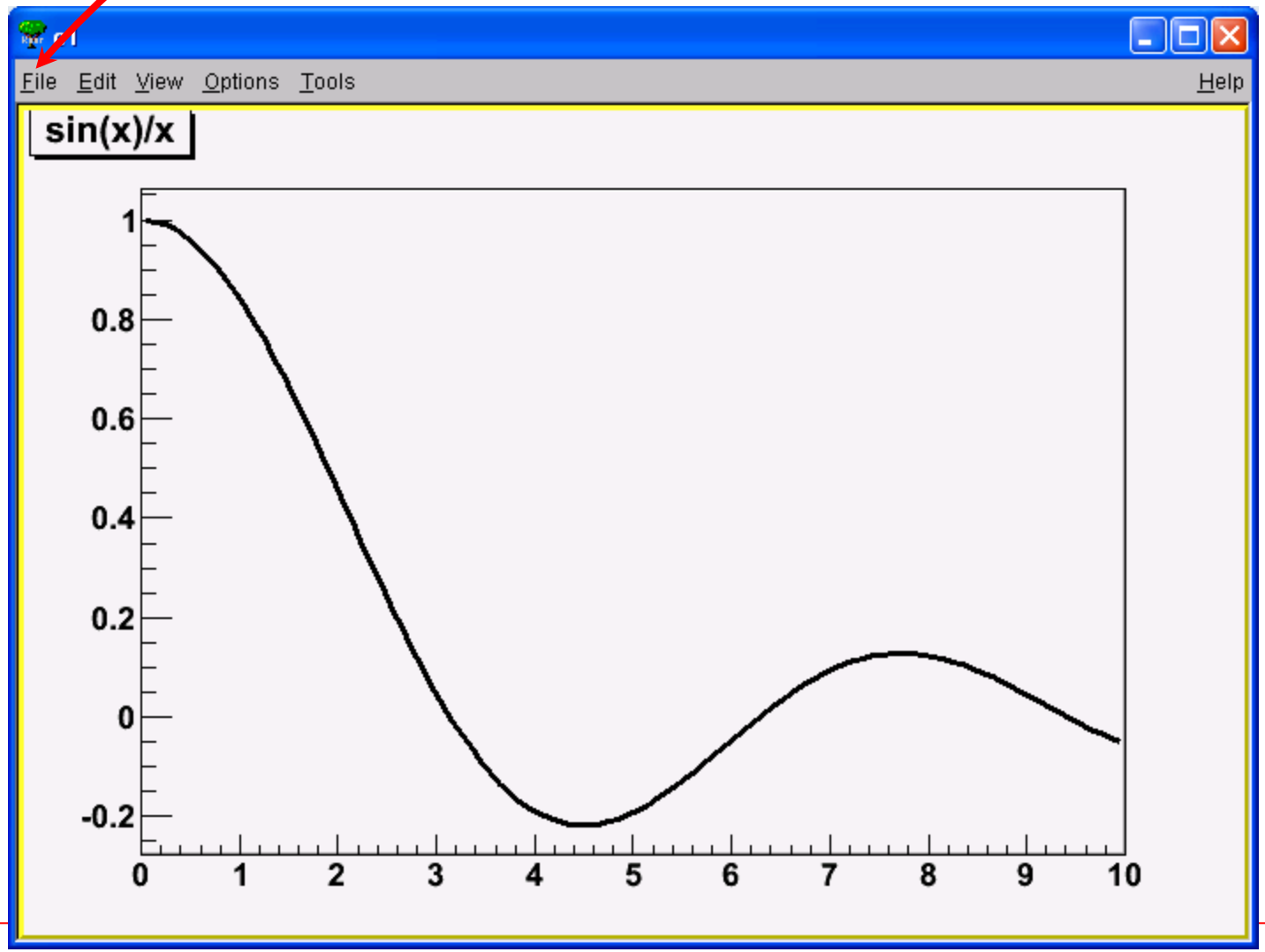
```
root [0] 15
(const int)15

root [1] 15.0
(const double)1.50000000000000000e+001

root [2] 15+12
(const int)27

root [3] 1 + sqrt(9.0)
(const double)4.00000000000000000e+000

root [4] for(int i=1; i<=5; i++) cout << "hello" << endl;

root [5] try up and down arrow to call back previous commands
```

Function Drawing

- Drawing sin(x)/x function between x = 0 and x = 10



Click here to save this graph for desired format



ROOT Macro Files

```
> edit myfile.C
```

```
{
   TF1 f("fun","sin(x)/x",0,10);
   f.Draw();
}
```

```
> root myfile.C
```

You can give a name to the macro.
Macro name has to be same name as the file!

```
void myfile()
{
   TF1 f("fun","sin(x)/x",0,10);
   f.Draw();
}
```

Loading / Executing Macros

```
> root
```

```
root [0] .L myfile.C   // load macro
root [1] .x myfile.C   // execute (run)
root [2] .q            // quit from root
```

Note that

unnamed macros can be executed by `.x` command

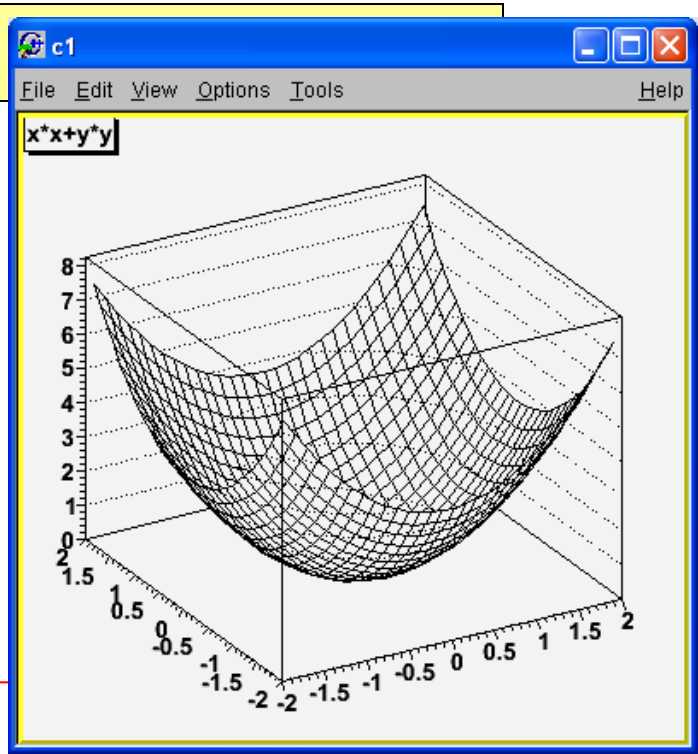
however they can not be loaded by `.L` command.

2D Functions

```
> edit fun2d.C
```

```
{
   TF2 f("fun", "x*x+y*y",-2,2, -2,2);
   f.Draw("surf");
}
```

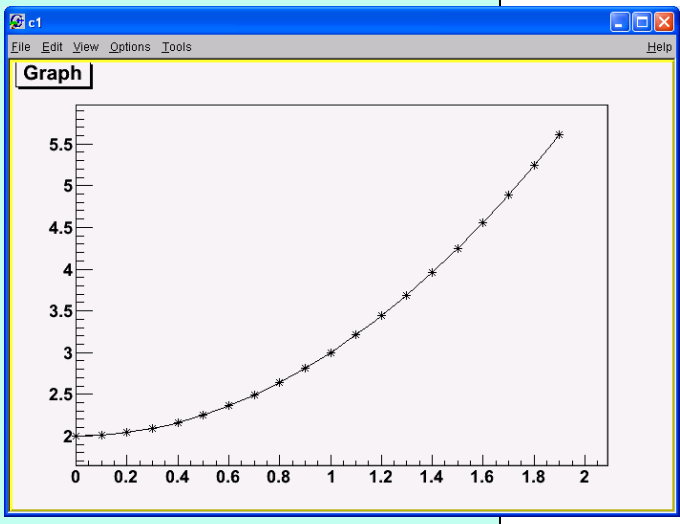
```
> root fun2d.C
```



Graphs

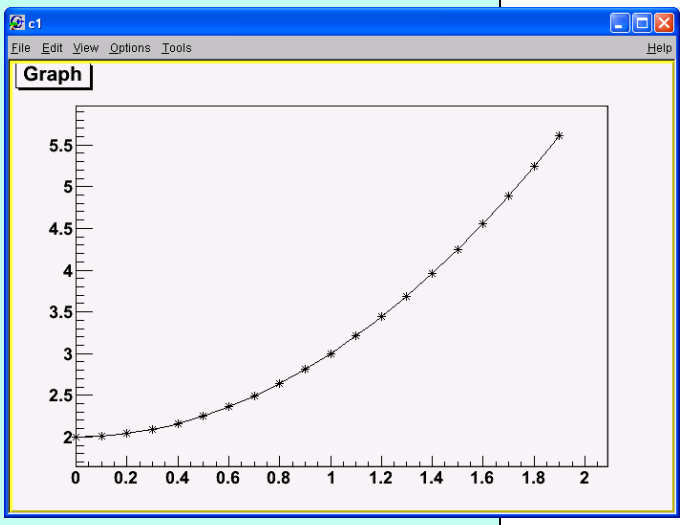
```
> edit grafik.C
```

```
{
   const int n = 20;
   double x[n], y[n];

   for(int i=0; i<20; i++){
      x[i] = i*0.1;
      y[i] = x[i]*x[i] + 2.0;
   }

   TGraph gr1(n, x, y);
   gr1.Draw("AC*");
}
```




```
> root grafik.C
```

```
{
   const int n = 20;
   double x[n], y[n];

   for(int i=0; i<20; i++){
      x[i] = i*0.1;
      y[i] = x[i]*x[i] + 2.0;
   }

   TGraph *gr1 = new TGraph(n, x, y);
   gr1->Draw("AC*");
}
```

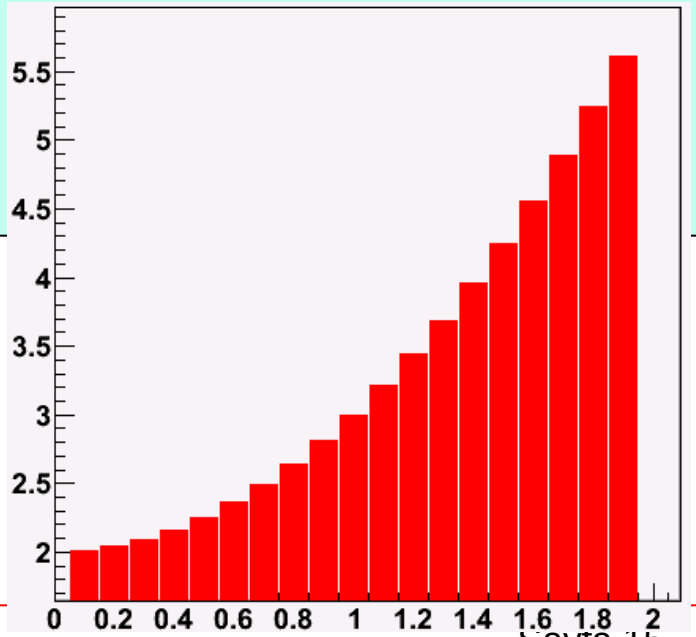


Some Graph Draw options

- `"L"` A simple poly-line between every points is drawn
- `"A"` Axis are drawn around the graph
- `"C"` A smooth curve is drawn
- `"*"` A star is plotted at each point
- `"P"` The current marker of the graph is plotted at each point
- `"B"` A bar chart is drawn at each point

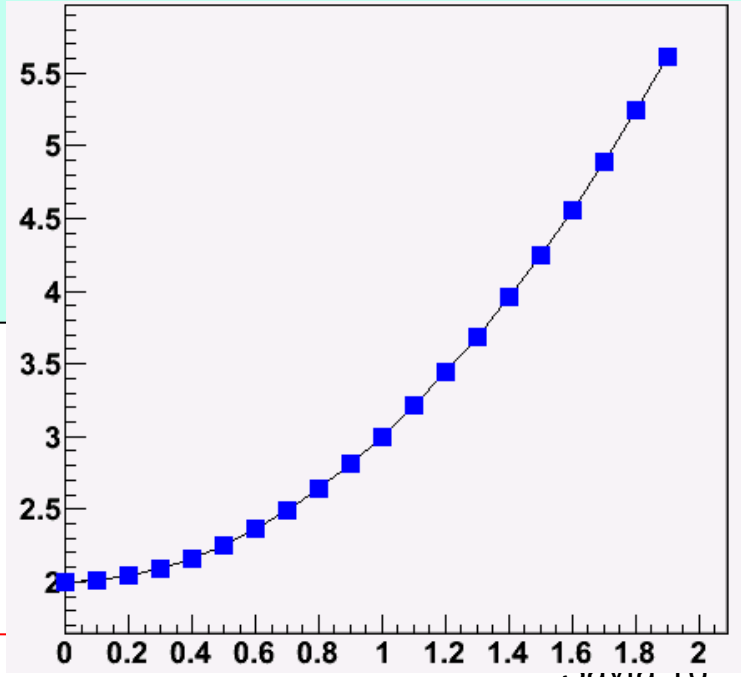
```
> edit grafik2.C
```

```
{
  const int n = 20;
  double x[n], y[n];
  for(int i=0;i<20; i++){
    x[i] = i*0.1;
    y[i] = x[i]*x[i] + 2.0;
  }

  TCanvas *c1 = new TCanvas("c1","ilk graf",200,10,500,500);
  TGraph *gr = new TGraph(n, x, y);

  gr->SetFillColor(kRed);
  gr->Draw("AB");
}
```



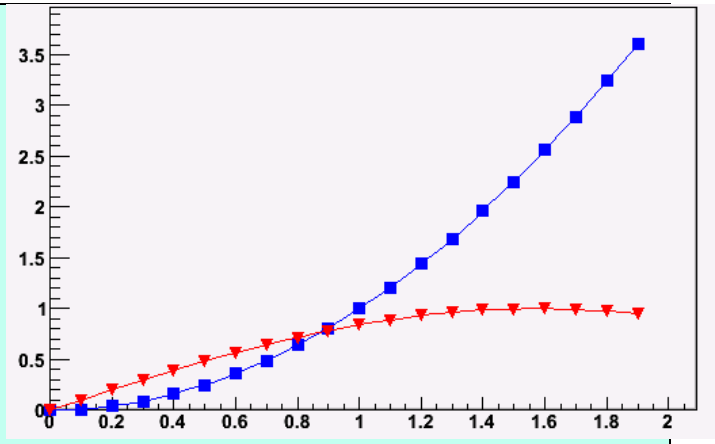
> edit grafik3.C

```
{
  const int n = 20;
  double x[n], y[n];
  for(int i=0;i<20; i++){
    x[i] = i*0.1;
    y[i] = x[i]*x[i] + 2.0;
  }

  TCanvas *c1 = new TCanvas("c1","ilk graf",200,10,500,500);
  TGraph *gr = new TGraph(n, x, y);
  gr->SetMarkerStyle(21);
  gr->SetMarkerColor(kBlue);
  gr->SetMarkerSize(1.3);
  gr->Draw("APL");
}
```

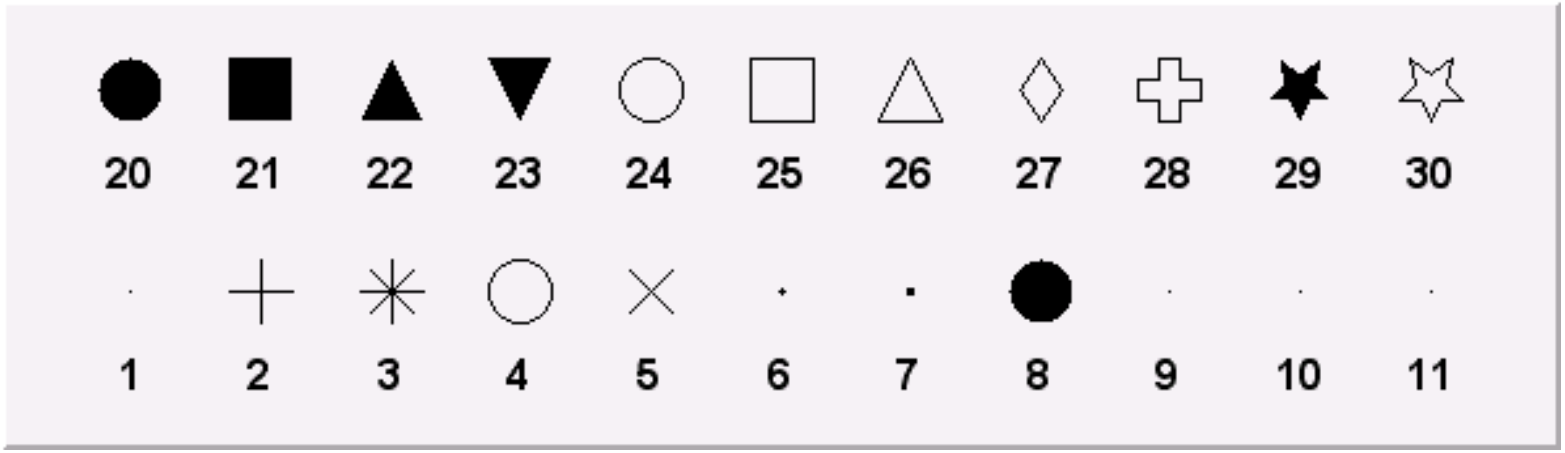


> edit grafik4.C

```
void grafik4()
{
  const Int_t n = 20;
  Double_t x[n], y1[n], y2[n];
  for (Int_t i=0; i<n; i++) {
    x[i]  = i*0.1;
    y1[i] = x[i]*x[i];
    y2[i] = sin(x[i]);
   }

   TGraph *gr1 = new TGraph(n, x, y1);
   TGraph *gr2 = new TGraph(n, x, y2);
   TCanvas *c1 = new TCanvas("c1","Two Graphs",200,10,600,400);

   gr1->SetMarkerStyle(21);
   gr1->SetMarkerColor(kBlue);
   gr1->SetLineColor(kBlue);
   gr1->Draw("ACP");
   // superimpose the second graph by leaving out the axis option "A"
   gr2->SetMarkerStyle(23);
   gr2->SetMarkerColor(kRed);
   gr2->SetLineColor(kRed);
   gr2->Draw("CP");
}
```



Marker Styles



Colors

Histograms

- Contain binned data
  probably the most important class in ROOT for the physicist
- Create a 1-dim Histo (float precision)

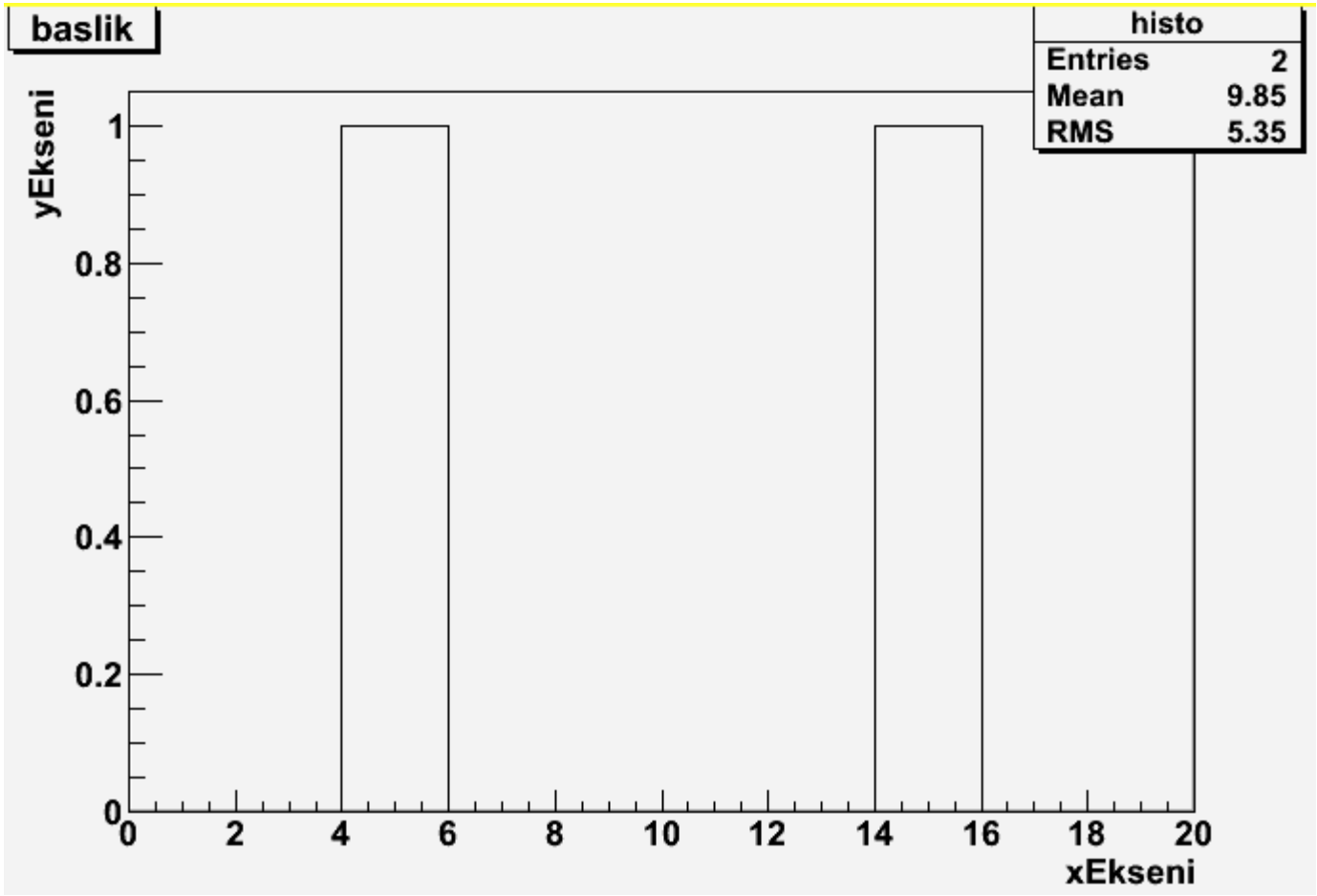
```
TH1F h("histo", "my histo; xtitle; ytitle", 20, 0 ,10);
```

  - `"histo"` is the name of the histogram
  - `"my histo; xtitle; ytitle"` are the title and x and y labels
  - 10 number of bins
  - 0, 20 limists on x-axis

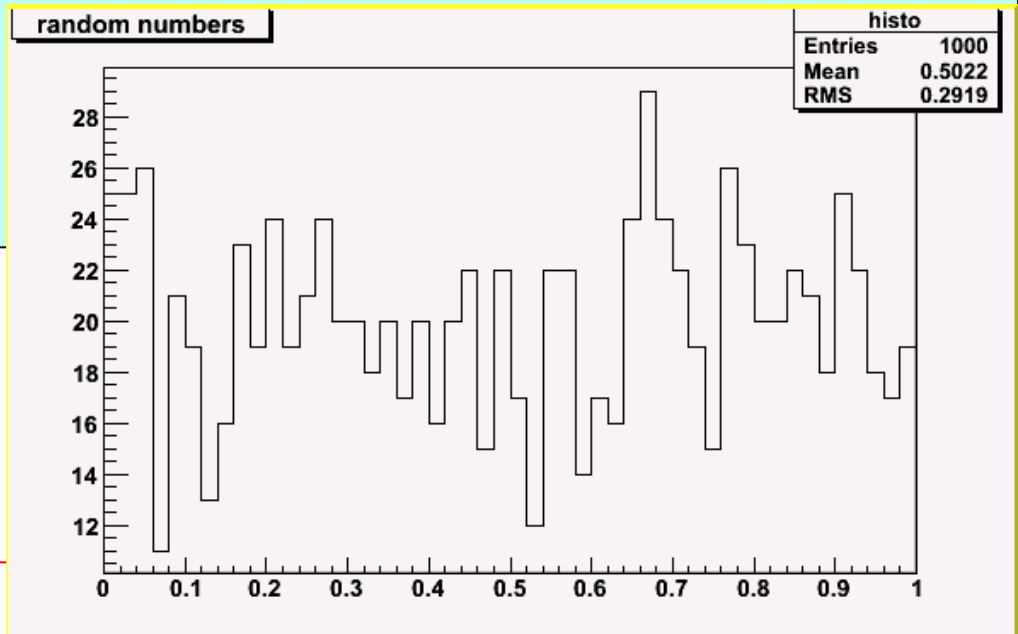
- Create a 1-dim Histo (double precision)

```
TH1D h("histo", "my histo; xtitle; ytitle", 20, 0 ,10);
```

```
root [0] TH1F h("histo","baslik;xEkseni;yEkseni",10,0,20)
root [1] h.Fill(4.5)
root [2] h.Fill(15.2)
root [3] h.Draw()
root [4]
```

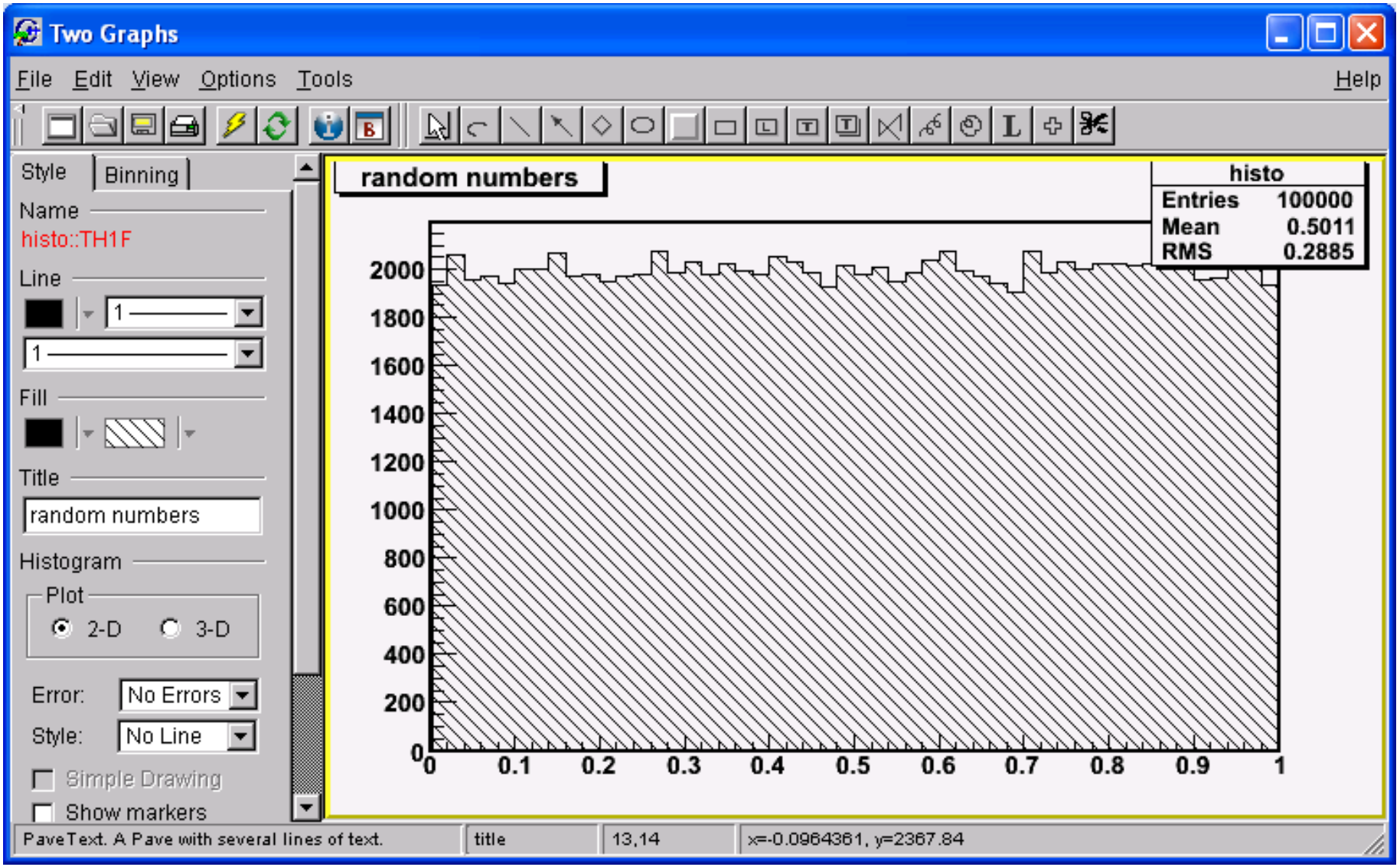


```
void histo1()
{
  gROOT->Reset();
  TCanvas *c1 = new TCanvas("c1","Histog",20,10,600,400);
  TH1F *h = new TH1F("histo","random numbers",50,0,1);

  int n = 1000;
  double r;

  for(Int_t i=1; i<=n; i++){
    r = gRandom->Uniform(); // a random # between [0,1]
    h->Fill(r);
  }

  h->Draw();
}
```



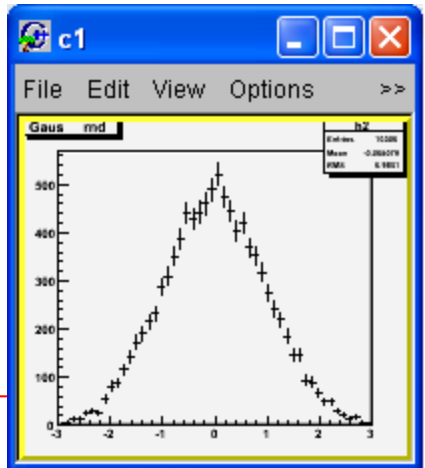
```
// Dynamic Plot
void histo2()
{
  gROOT->Reset();
  TCanvas *c1 = new  TCanvas("c1","Histog",20,10,600,400);
  TH1F *h = new TH1F("histo","random numbers",50,0,1);

  h->Draw();
  h->SetMinimum(0);
  int n = 100000;

  double r;
  for(Int_t i=1; i<=n; i++){
    r = gRandom->Uniform(); // a random # between [0,1]
    h->Fill(r);
    if(i%100==0) {
      c1->Modified();
      c1->Update();
    }
  }
}
```

Using GUI


Two Graphs
File Edit View Options Tools Help
Style Binning
Name
histo::TH1F
Line
Fill
Title
random numbers
Histogram
Plot
2-D 3-D
Error: No Errors
Style: No Line
Simple Drawing
Show markers
random numbers
histo
Entries 100000
Mean 0.5011
RMS 0.2885
2000
1800
1600
1400
1200
1000
800
600
400
200
0
0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1
PaveText. A Pave with several lines of text.
title
13,14
x=-0.0964361, y=2367.84

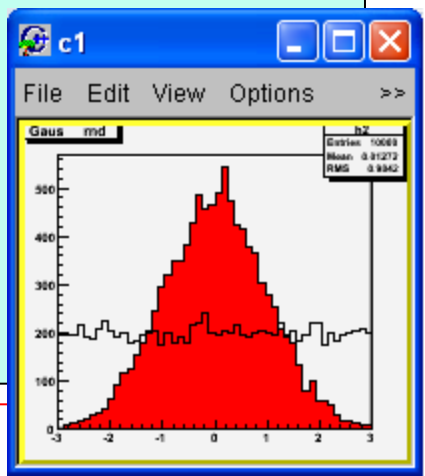
Drawing Errobars

```
// Error bars
void histo3()
{
  TCanvas *c1 = new TCanvas("c1","c1",10,10,500,500);
  TH1F    *h1 = new TH1F("h2","Gaus rnd",50, -3, 3);

  for(int i=0; i<10000; i++){
    h1->Fill( gRandom->Gaus(0,1) );
  }

  h1->Draw("simple");
}
```



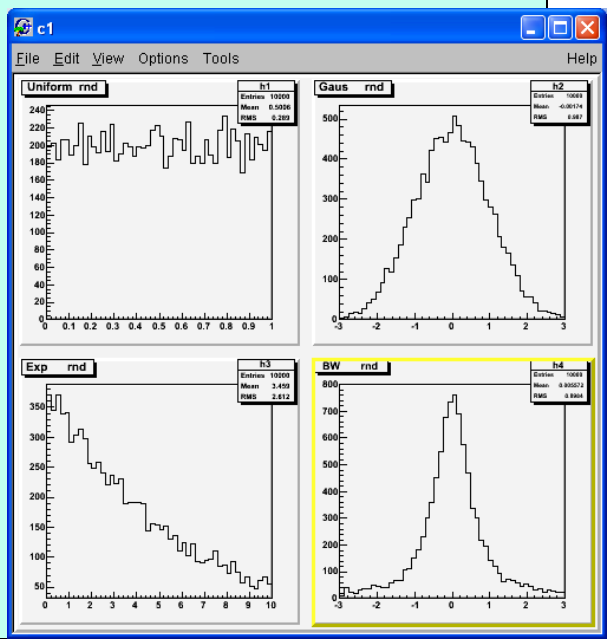
Superimposing Histograms

```
// superimpose two histograms
void histo4()
{
  TCanvas *c1 = new TCanvas("c1","c1",10,10,500,500);
  TH1F    *h1 = new TH1F("h1","Uniform rnd",50, -3, 3);
  TH1F    *h2 = new TH1F("h2","Gaus    rnd",50, -3, 3);
  h1->SetMinimum(0);

  for(int i=0; i<10000; i++){
    h1->Fill( gRandom->Uniform(-3,3) );
    h2->Fill( gRandom->Gaus(0,1) );
  }

  h2->SetFillColor(2);

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



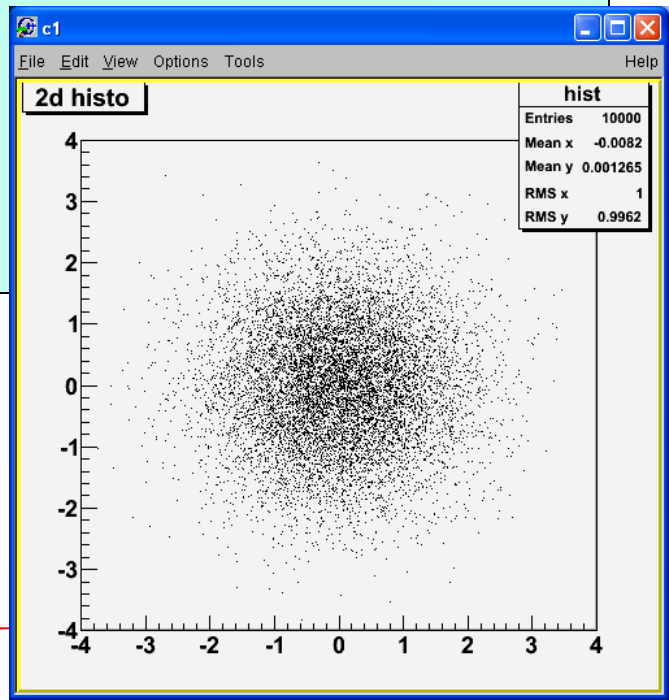
Dividing Canvas

```
void histo5()
{
  TCanvas *c1 = new TCanvas("c1","c1",10,10,500,500);
  TH1F    *h1 = new TH1F("h1","Uniform  rnd",50, 0.0, 1.0);
  TH1F    *h2 = new TH1F("h2","Gaus     rnd",50,-3.0, 3.0);
  TH1F    *h3 = new TH1F("h3","Exp      rnd",50, 0.0,10.0);
  TH1F    *h4 = new TH1F("h4","BW       rnd",50,-3.0, 3.0);
  h1->SetMinimum(0);

  for(int i=0; i<10000; i++){
    h1->Fill( gRandom->Uniform() );
    h2->Fill( gRandom->Gaus(0,1) );
    h3->Fill( gRandom->Exp(5) );
    h4->Fill( gRandom->BreitWigner(0,1) );
  }

  c1->Divide(2,2);
  c1->cd(1); h1->Draw();
  c1->cd(2); h2->Draw();
  c1->cd(3); h3->Draw();
  c1->cd(4); h4->Draw();
}
```



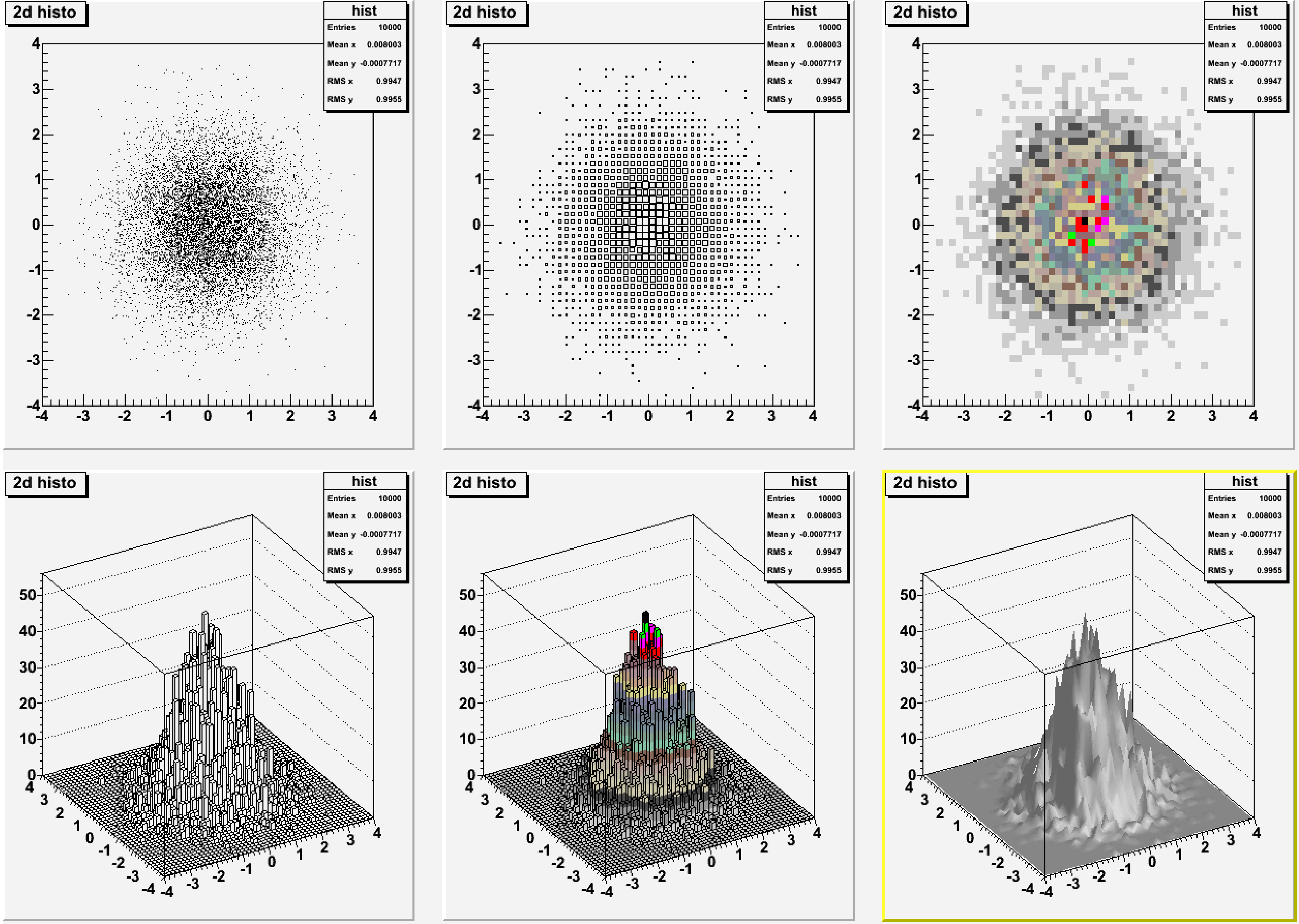
2D Histograms

```
void histo6()
{
  TCanvas *c1 = new TCanvas("c1","c1",10,10,500,500);
  TH2F    *h  = new TH2F("hist","2d histo",
                         50,-4,4, 50,-4,4);

  for(int i=0; i<10000; i++){
    double r1 = gRandom->Gaus();
    double r2 = gRandom->Gaus();
    h->Fill(r1, r2);
  }

  h->Draw();
}
```



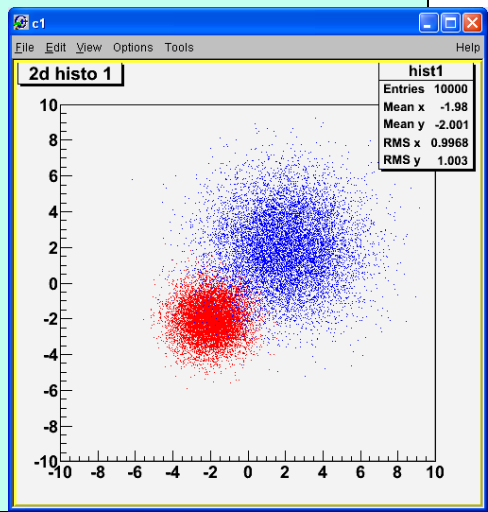
2D Histograms on Subpads

```
void histo7()
{
  TCanvas *c1 = new TCanvas("c1","c1",10,10,500,500);
  TH2F    *h  = new TH2F("hist","2d histo",
                         50,-4,4, 50,-4,4);

  for(int i=0; i<10000; i++){
    double r1 = gRandom->Gaus();
    double r2 = gRandom->Gaus();
    h->Fill(r1, r2);
  }

  c1->Divide(3,2);

  c1->cd(1); h->Draw();          // default: scatter plot
  c1->cd(2); h->Draw("box");     // box plot
  c1->cd(3); h->Draw("col");     // colored

  c1->cd(4); h->Draw("lego");    // lego plot
  c1->cd(5); h->Draw("lego2");   // colored lego plot
  c1->cd(6); h->Draw("surf4");   // surface plot
}
```


2d histo
hist
Entries 10000
Mean x 0.008003
Mean y -0.0007717
RMS x 0.9947
RMS y 0.9955

Superimposing two 2D Histograms

```
void histo8()
{
  TCanvas *c1 = new TCanvas("c1","c1",10,10,500,500);
  TH2F    *h1 = new TH2F("hist1","2d histo 1",
                         50,-10,10, 50,-10,10);
  TH2F    *h2 = new TH2F("hist2","2d histo 2",
                         50,-10,10, 50,-10,10);

  for(int i=0; i<10000; i++){
    double r1 = gRandom->Gaus(-2,1);   // mean=-2, sigma=1
    double r2 = gRandom->Gaus(-2,1);
    h1->Fill(r1, r2);
    double r3 = gRandom->Gaus(2,2);    // mean=2, sigma=2
    double r4 = gRandom->Gaus(2,2);
    h2->Fill(r3, r4);
  }

  h1->SetMarkerColor(kRed);
  h2->SetMarkerColor(kBlue);
  h1->Draw();
  h2->Draw("same");
}
```

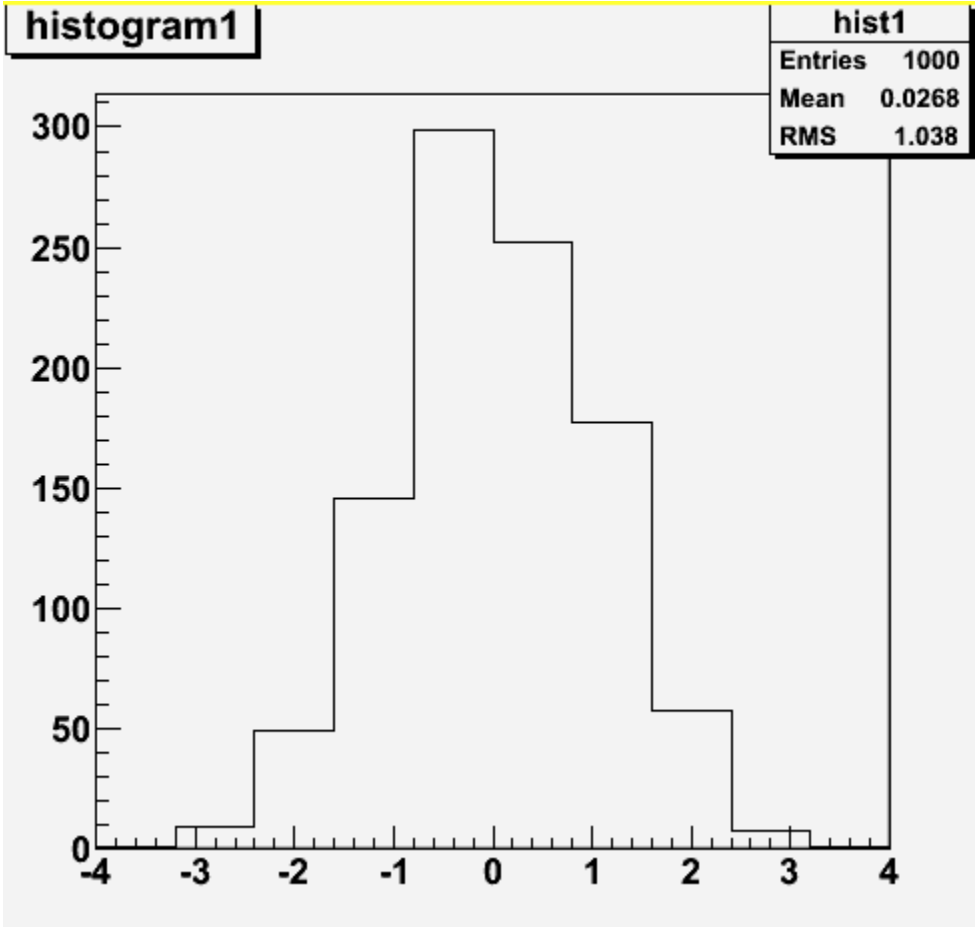


Getting Histogram Bin Content

```
void histo9()
{
  TCanvas *c1 = new TCanvas("c1","c1",10,10,500,500);
  TH1F    *h1 = new TH1F("hist1","histogram1",10,-4,4);

  for(int i=0; i<1000; i++){
    double r = gRandom->Gaus(); // mean=0, sigma=1
    h1->Fill(r);
  }

  h1->Draw();
  for(int i=1; i<=10; i++){
    cout << h1->GetBinContent(i) << endl;
  }

  cout << "mean= " << h1->GetMean() << endl;
  cout << "RMS = " << h1->GetRMS() << endl;
}
```

Output

```
1
9
49
146
299
252
177
58
1

mean= -0.335073
RMS = 0.984885
```



Setting Histogram Bin Content

```
void histo10()
{
  TCanvas *c1 = new TCanvas("c1","c1",10,10,500,500);
  TH1F    *h1 = new TH1F("hist1","histogram1",10,0,5);

  h1->SetBinContent(1,64);
  h1->SetBinContent(2,32);
  h1->SetBinContent(3,16);
  h1->SetBinContent(4,8);
  h1->SetBinContent(5,4);
  h1->SetBinContent(6,2);
  h1->SetBinContent(7,1);

  h1->Draw();

  cout << "mean= " << h1->GetMean() << endl;
  cout << "RMS = " << h1->GetRMS() << endl;
}
```

Output

```
mean= 0.702381
RMS = 0.595714
```