Introduction to Root program:L3

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.

What do we learn from this Macro

How to	Commands

Create a Root file ?	TFile *f = new TFile("basic.root","RECREATE");
	//option: NEW, CREATE, RECREATE, UPDATE, or READ
	//Book and fill histograms and trees
	//------
	f->Write(); //write the file
	f->Close(); //close the file

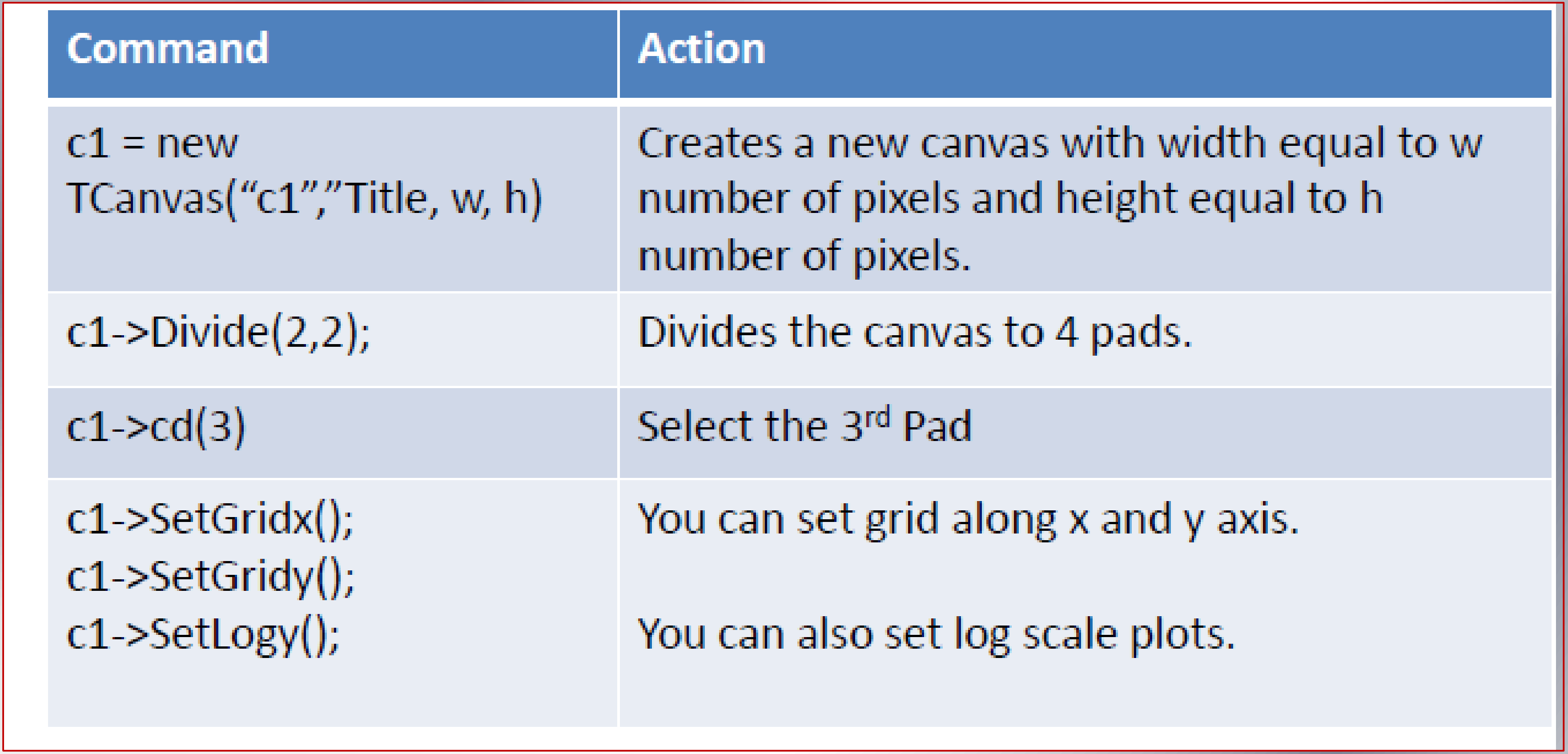
Book and fill a histogram ?	TH1F *h1 = new TH1F("h1","x distribution",100,-4,4);
	
	/*do some calculation and get the parameter that you want to fill*/
	
	h1->Fill(x);

Book and fill a tree ?	TNtuple *ntuple = new TNtuple("ntuple","data from ascii file","x:y:z"); /*do some calculation and get the parameter that you want to fill*/ ntuple->Fill(x,y,z);
CINT Data types	Int_t and Float_t (see http://root.cern.ch/root/html/ListOfTypes.html)

Histograms drawing options

" SAME": Superimpose on previous picture in the same pad.
" CYL": Use cylindrical coordinates.
" POL": Use polar coordinates.
" SPH": Use spherical coordinates.
" PSR": Use pseudo-rapidity/phi coordinates.
" LEGO": Draw a lego plot with hidden line removal.
" LEGO1": Draw a lego plot with hidden surface removal.
" LEGO2": Draw a lego plot using colors to show the cell contents.
" SURF": Draw a surface plot with hidden line removal.
" SURF1": Draw a surface plot with hidden surface removal.
" SURF2": Draw a surface plot using colors to show the cell contents.
" SURF3": Same as SURF with a contour view on the top.
" SURF4": Draw a surface plot using Gouraud shading.
“ SURF5”: Same as SURF3 but only the colored contour is drawn.

Canvas: An area mapped to a window

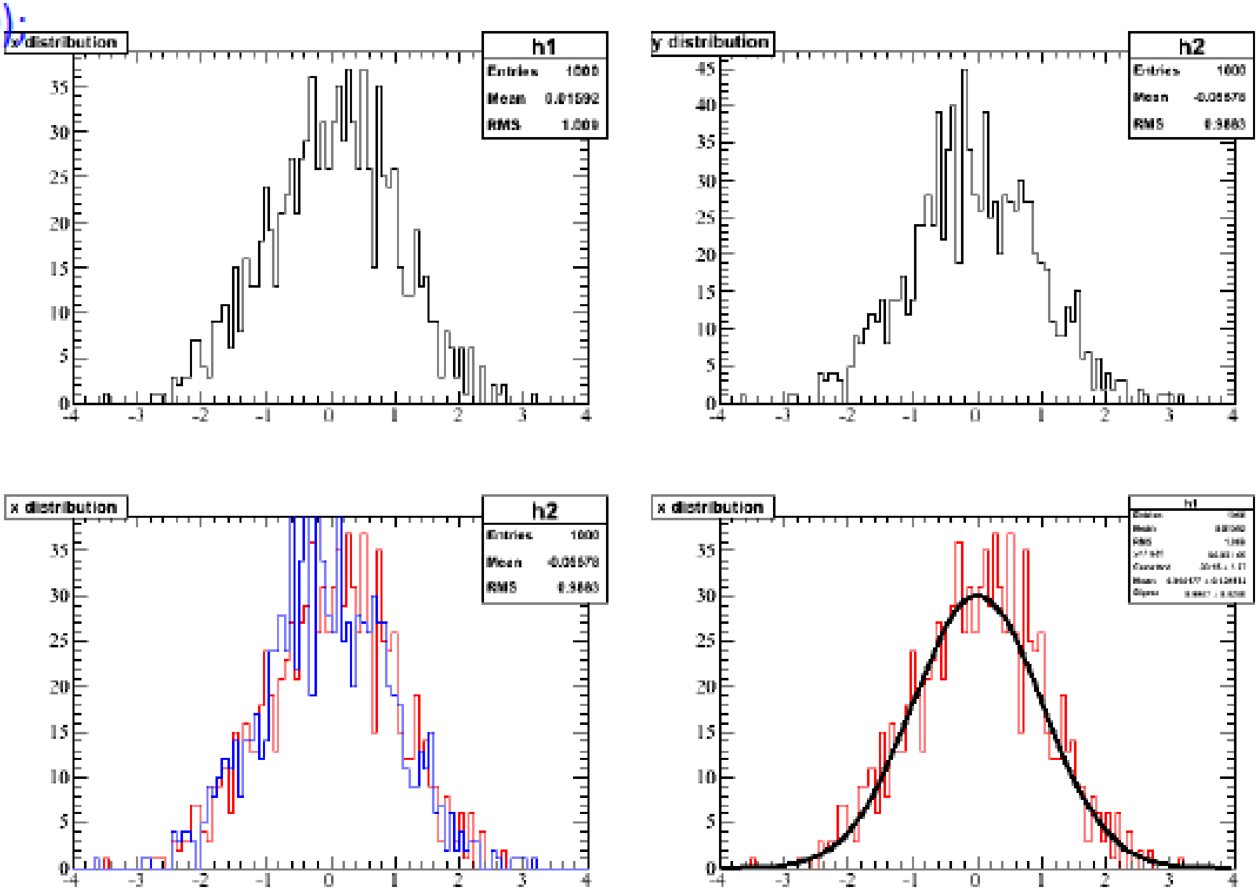


Command	Action
c1 = new TCanvas(“c1”,”Title, w, h)	Creates a new canvas with width equal to w number of pixels and height equal to h number of pixels.
c1->Divide(2,2);	Divides the canvas to 4 pads.
c1->cd(3)	Select the 3rd Pad

c1->SetGridx();
c1->SetGridy();
c1->SetLogy();
	You can set grid along x and y axis.
	You can also set log scale plots.

Canvas: Example

```
root [1] c1 = new
TCanvas("c1","Title",800,600);
root [2] c1->Divide(2,2);
root [3] c1->cd(1);
root [4] h1->Draw();
root [5] c1->cd(2);
root [6] h2->Draw();
root [7] c1->cd(3);
root [8] h1->SetLineColor(2)
root [9] h2->SetLineColor(4)
root [10] h1->Draw();
root [11] h2->Draw("same");
root [12] c1->cd(4);
root [13] h1->Fit("gaus");
```

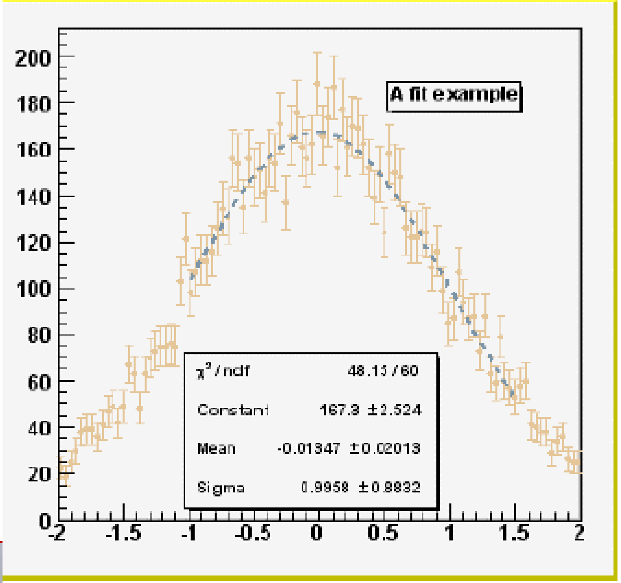


Which fitting functions ?

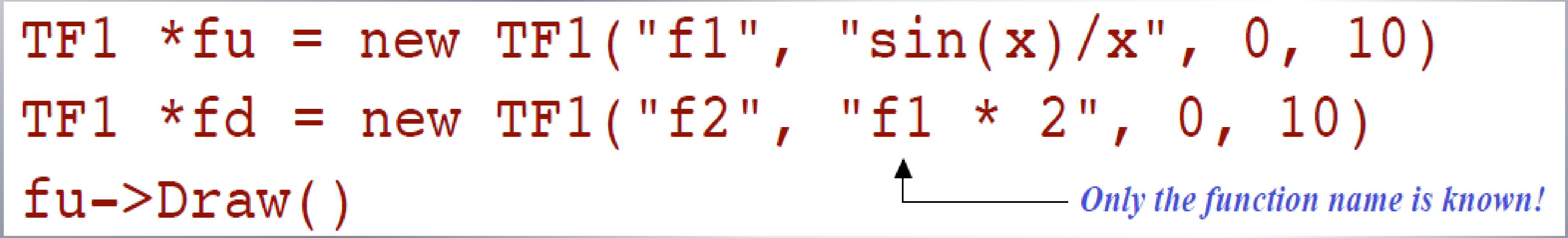
- The predefined functions:
  - "gaus" = p0*exp(-0.5*pow((x-p1)/p2),2)
  - "expo" = exp(p0+p1*x)
  - "polN" = p0 + p1*x + p2*pow(x,2) + p3*...
  - "landau" (guess the formula!)

How to obtain the values of the fit parameters ?

```
TF1 *gfit = (TF1 *)h->GetFunction("gaus")
gfit->GetParameter(0)
gfit->GetParameter(1) ...
gfit->GetParError(0) ...
double par[3]
gfit->GetParameters(par)
```



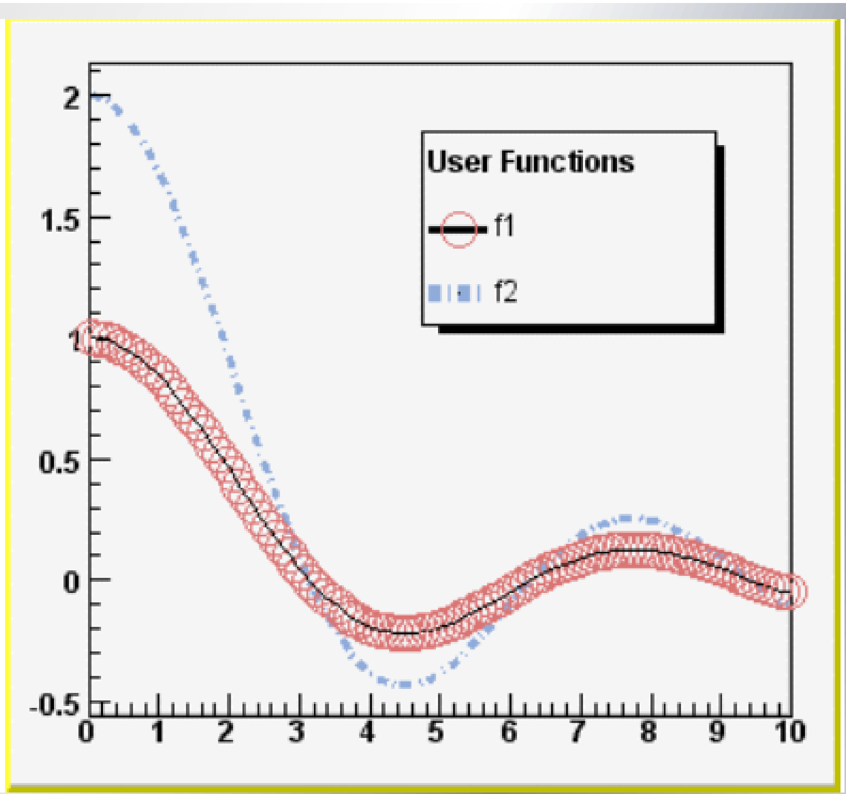
Creating a user defined function



```
TF1 *fu = new TF1("f1", "sin(x)/x", 0, 10)
TF1 *fd = new TF1("f2", "f1 * 2", 0, 10)
fu->Draw()
```

Only the function name is known!

```
fd->Draw("same")
```

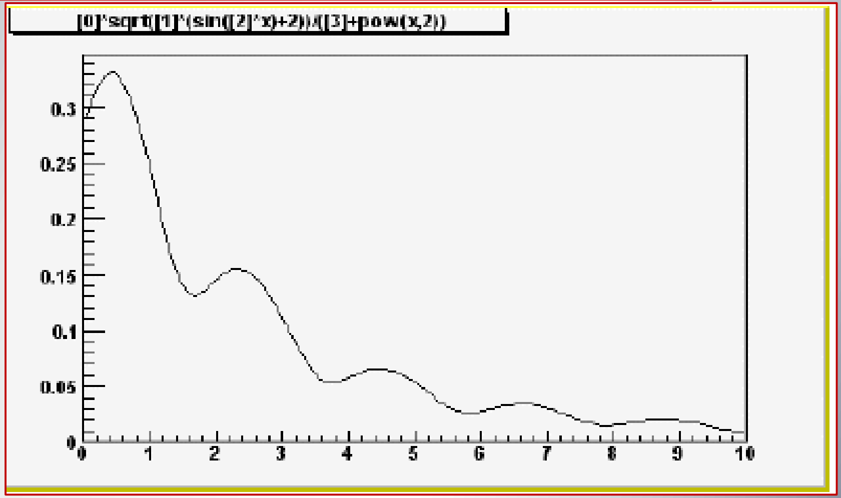


Including Parameters

```
TF1 *ft = new TF1("f3", "[0]*sqrt([1]*(sin([2]*x)+2))
   /([3]+pow(x,2))", 0, 10)
```

```
ft->SetParameters(1,1,3,5)
ft->Draw()
```

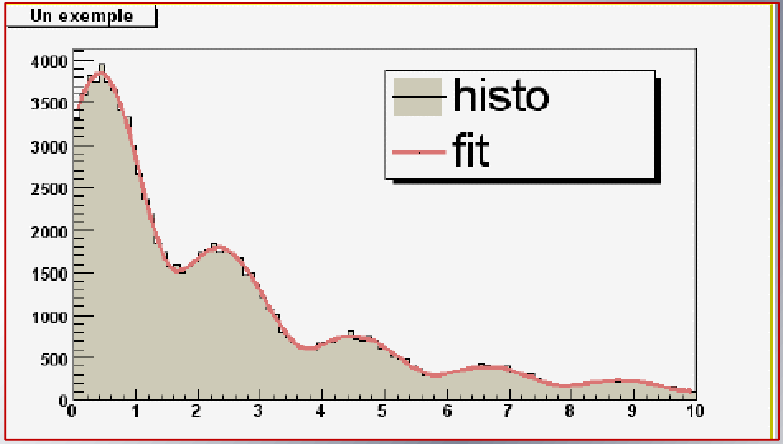
index	0	1	2	3
content	1	1	3	5



```
TH1F *hd = new TH1F("h2","Un exemple", 100, 0, 10)
```

```
hd->FillRandom("f3",100000)
ft->SetParameters
    (h2->GetMaximum(),1,2.8,6.)
hd->Fit("f3")
```

index	0	1	2	3
content	h2->GetMaximum()	1	2.8	6



ROOT Trees

- Store large quantities of same-class objects
- TTree class is optimized to reduce disk space and enhance access speed
- TTree can hold all kind of data
- TNtupleis a Ttree that is limited to only hold floating-point numbers

If we do not use TTree, we need to

- ✓ read each event in its entirety into memory
- ✓ extract the parameters from the event
- ✓ Compute quantities from the same
- ✓ fill a histogram

Create Trees/TNtuple

- Tfile *F = new Tfile(“test.root”,RECREATE);
- TTree *T = new TTree("T","test");
- T->Branch("x",&x,"x/F");
- T->Branch("y",&y,"x/F");
- T->Branch("z",&z,"x/F");

// Read/or calculate x,y and z

- T->Fill();
- T->Close();
- F->Close();

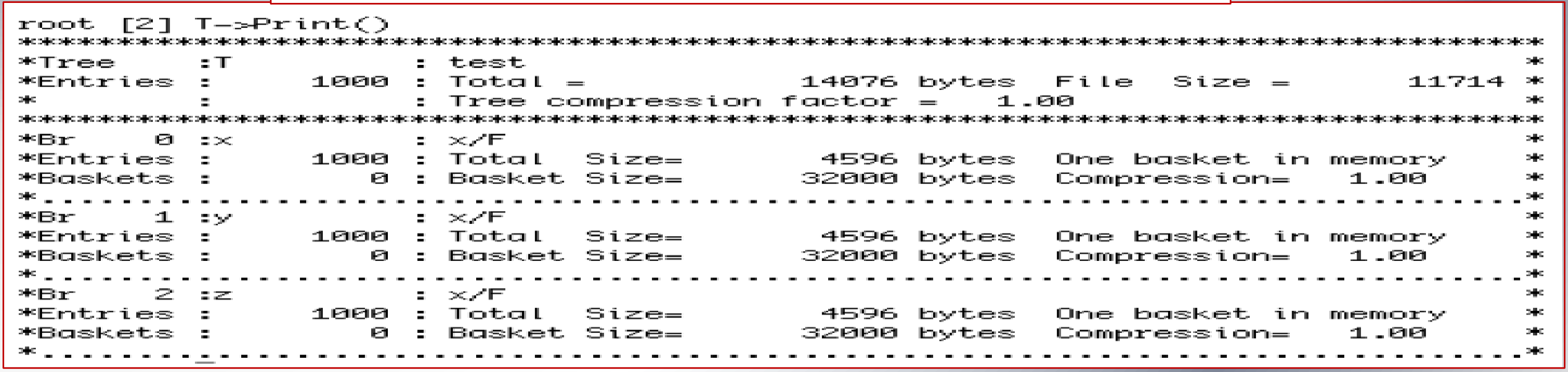
- Tfile *F = new Tfile(“test.root”,RECREATE);
- TNtuple *T = new TNtuple("ntuple","data from ascii file","x:y:z");

// Read/or calculate x,y and z

- T->Fill(x,y,z);
- T->Close();
- F->Close();

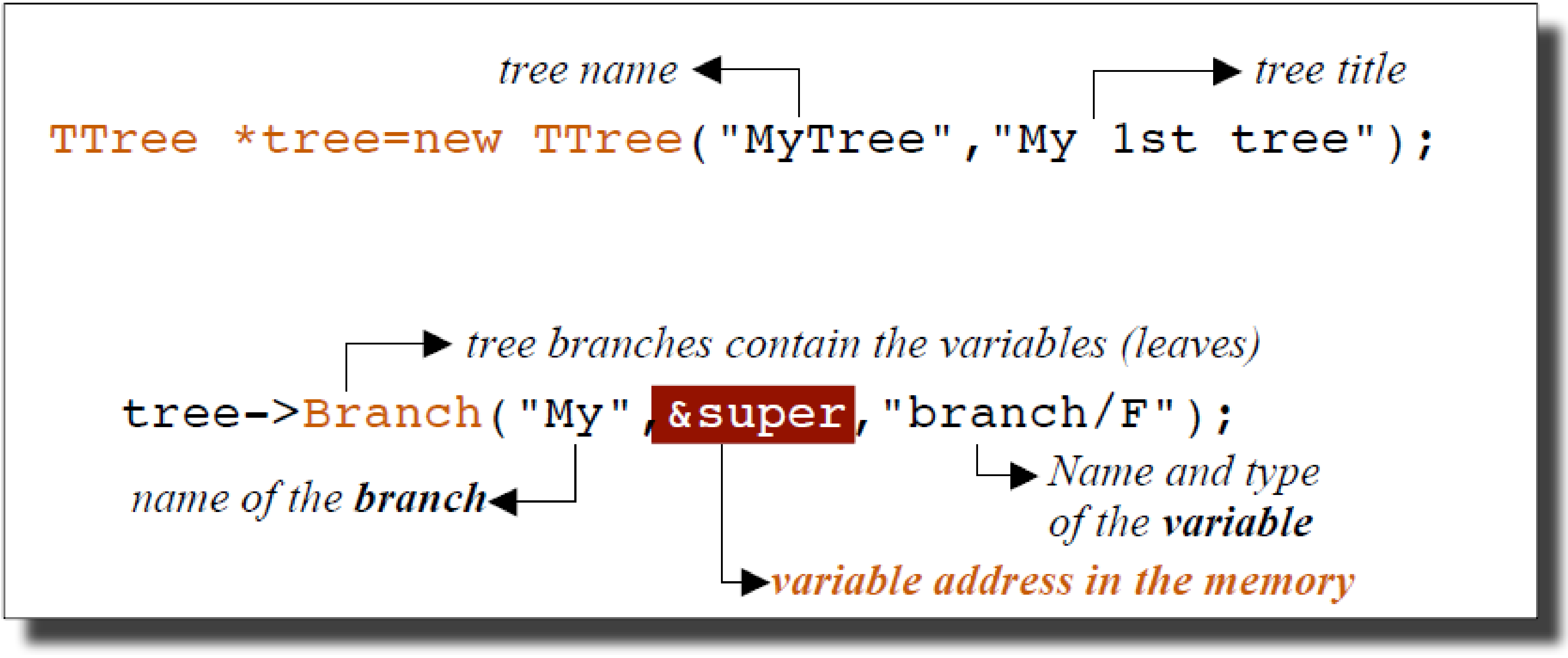
Draw: T->Draw(“x”);

T -> Print(); //print contents of root file



```
root [2] T->Print()
******************************************************************************
*Tree    :T         : test                                                   *
*Entries :     1000 : Total =           14076 bytes  File  Size =      11714 *
*        :          : Tree compression factor =   1.00                       *
******************************************************************************
*Br    0 :x         : x/F                                                    *
*Entries :     1000 : Total  Size=       4596 bytes  One basket in memory    *
*Baskets :        0 : Basket Size=      32000 bytes  Compression=   1.00     *
*............................................................................*
*Br    1 :y         : x/F                                                    *
*Entries :     1000 : Total  Size=       4596 bytes  One basket in memory    *
*Baskets :        0 : Basket Size=      32000 bytes  Compression=   1.00     *
*............................................................................*
*Br    2 :z         : x/F                                                    *
*Entries :     1000 : Total  Size=       4596 bytes  One basket in memory    *
*Baskets :        0 : Basket Size=      32000 bytes  Compression=   1.00     *
*............................................................................*
```

- A TTree can contain integers, real numbers, *structures*, even *objects*...



tree name

tree title

```
TTree *tree=new TTree("MyTree","My 1st tree");
```

tree branches contain the variables (leaves)

```
tree->Branch("My",&super,"branch/F");
```

name of the ***branch***

Name and type of the ***variable***

variable address in the memory

- Simple variables

```
Int_t mult;
tree->Branch("anInteger", &mult, "Mult/I");
Double_t ToF;
tree->Branch("aDouble", &ToF, "TdV/D");
```

- Fixed size array

```
Double_t Z[50];
tree->Branch("Z_branch", Z, "Charge[50]/D");
```

Beware!! The array name = the array address !!

- Variable size array

```
tree->Branch("Mult", &mult, "mult/I");
tree->Branch("dM/dZ", Z, "Z[mult]/D");
```

Useful command in using tree

Command	Action
T->Print();	Prints the content of the tree
T->Scan();	Scans the rows and columns
T->Draw(“x”);	Draw a branch of tree

How to apply cuts:
T->Draw(“x”,”x>0”);
T->Draw(“x”,”x>0 && y>0”);
	Draw “x” when “x>0”
	Draw “x” when both x >0 and y >0

T->Draw(“y”,” “,”same”);	Superimpose “y” on “x”
T->Draw(“y:x”);	Make “y vs x” 2d scatter plot
T->Draw(“z:y:x”);	Make “z:y:x” 3d plot
T->Draw(“sqrt(x*x+y*y)”);	Plot calculated quantity
T->Draw(“x>>h1”);	Dump a root branch to a histogram

Thanks!