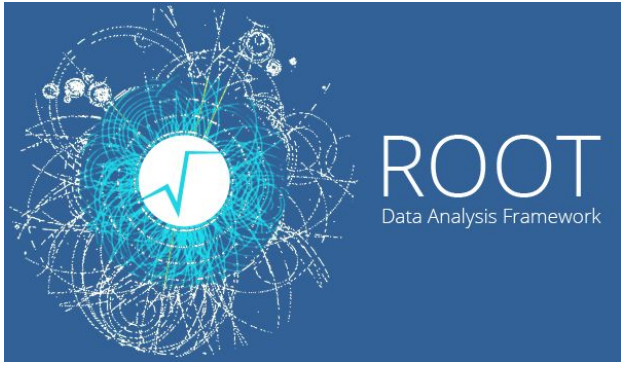
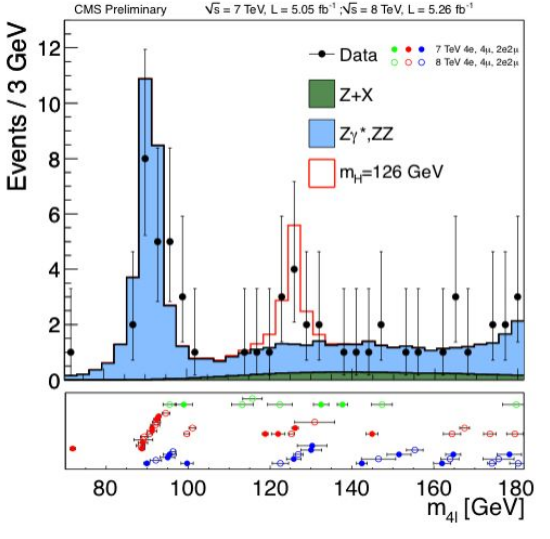
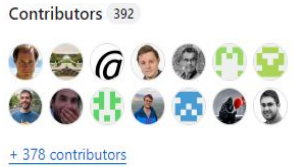
Introdução ao ROOT



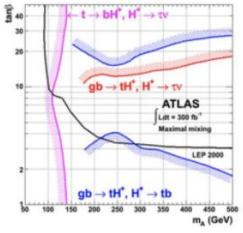


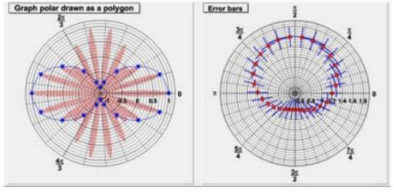
Introdução

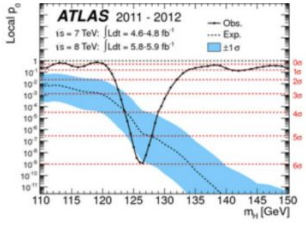
- Principal ferramenta de análise de dados usado nos experimentos do CERN
- É um programa de computador e biblioteca orientada a objetos desenvolvido pelo CERN (René Brun, Fons Rademakers) na década de 90
  - e em contínuo desenvolvimento
  - versão atual v6.30.04 (fev. 2024)
- Usado para:
  - histogramas, gráficos, funções, fits, ferramentas estatísticas, ferramentas matemáticas em geral, interpretados de C++ e python, manipulação de dados, geração de números aleatórios, e muito muito mais

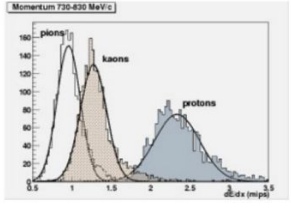


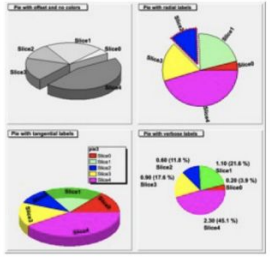
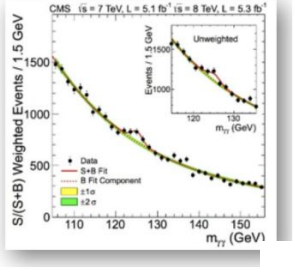
root.cern







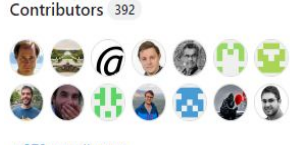


usado nos experimentos

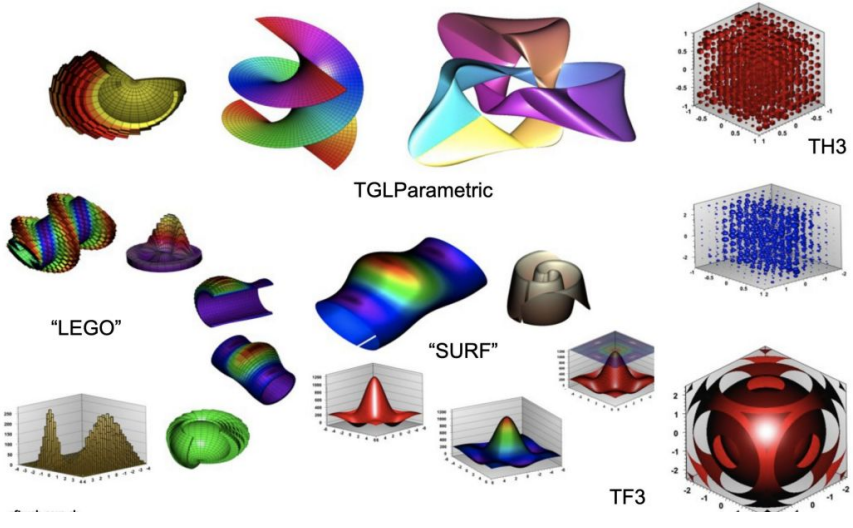
ca orientada a objetos

ns Rademakers) na década

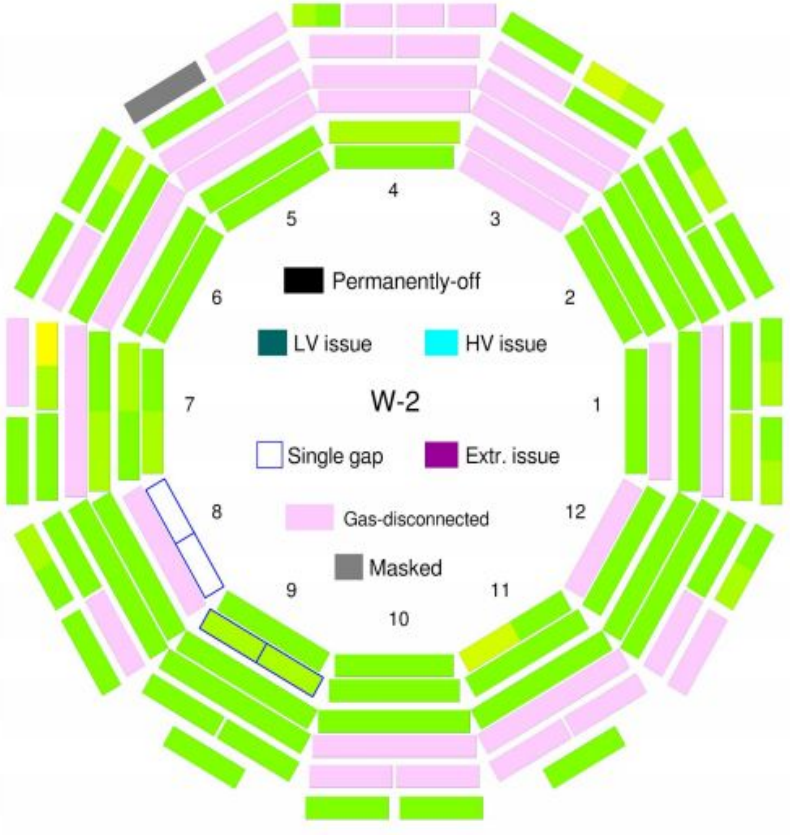


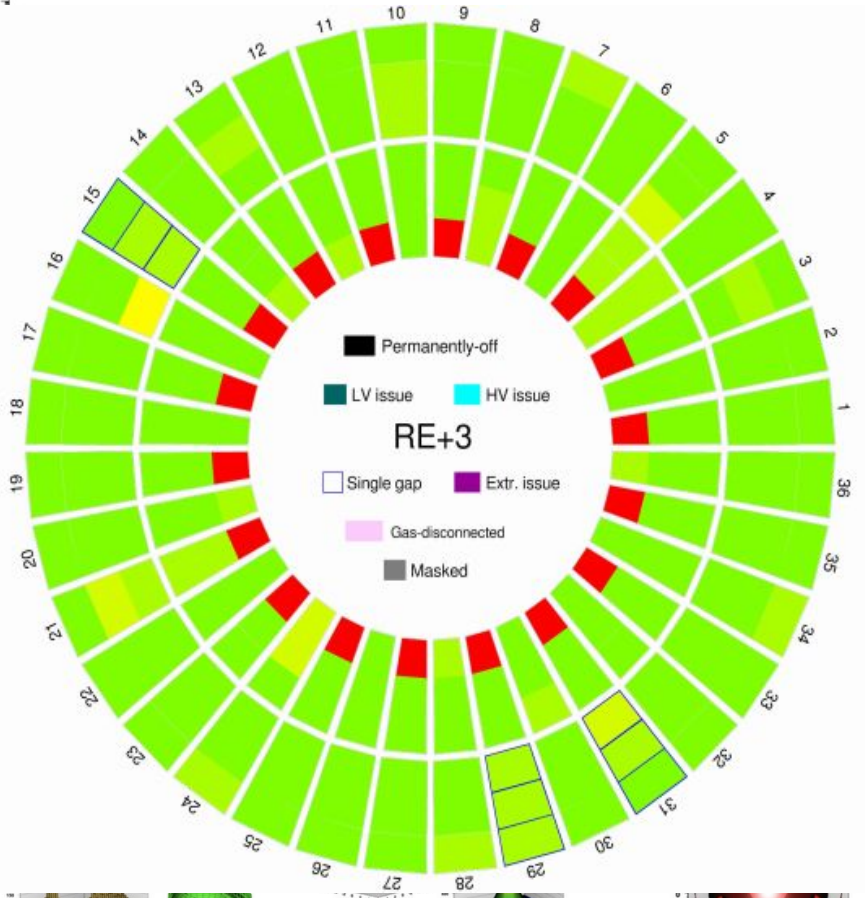
- e em contínuo desenvolvimento
- versão atual v6.30.04 (fev. 2024)

- Usado para:
  - histogramas, gráficos, funções, fits, ferramer
    matemáticas em geral, interpretados de C++
    geração de números aleatórios, e muito muit



root.cer





rrame
de C++
to mui

root.cer

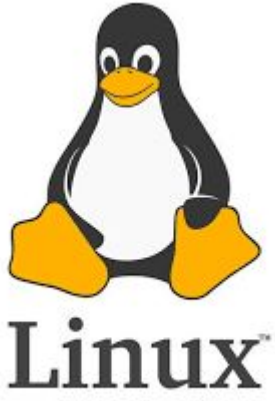
TF3

Introdução

Links importantes:

- https://root.cern/ - Pagina oficial
- https://root.cern/doc/master/index.html - Guia com todas as referências ao ROOT
- https://root-forum.cern.ch - Fórum
- https://root.cern/ROOTPrimer - Tutorial para os iniciantes

Introdução



Instalação -> https://root.cern/install/

Pode ser um pouco estressante…

Introdução

Installing ROOT from source

- You can install the ROOT's sources from the download area or using directly the Git repository.
- Install using Git repository:

Clone the repo

```
$ git clone https://github.com/root-project/root.git
```

Make a directory for building

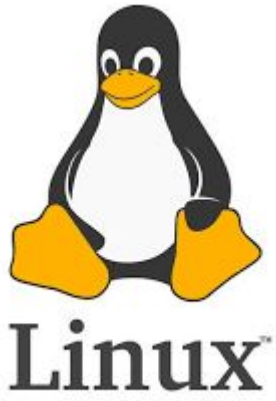
```
$ mkdir build
$ cd build
```

Run cmake and make

```
$ cmake ../root
$ make -j8
```

Setup and run ROOT

```
$ source bin/thisroot.sh
$ root
```



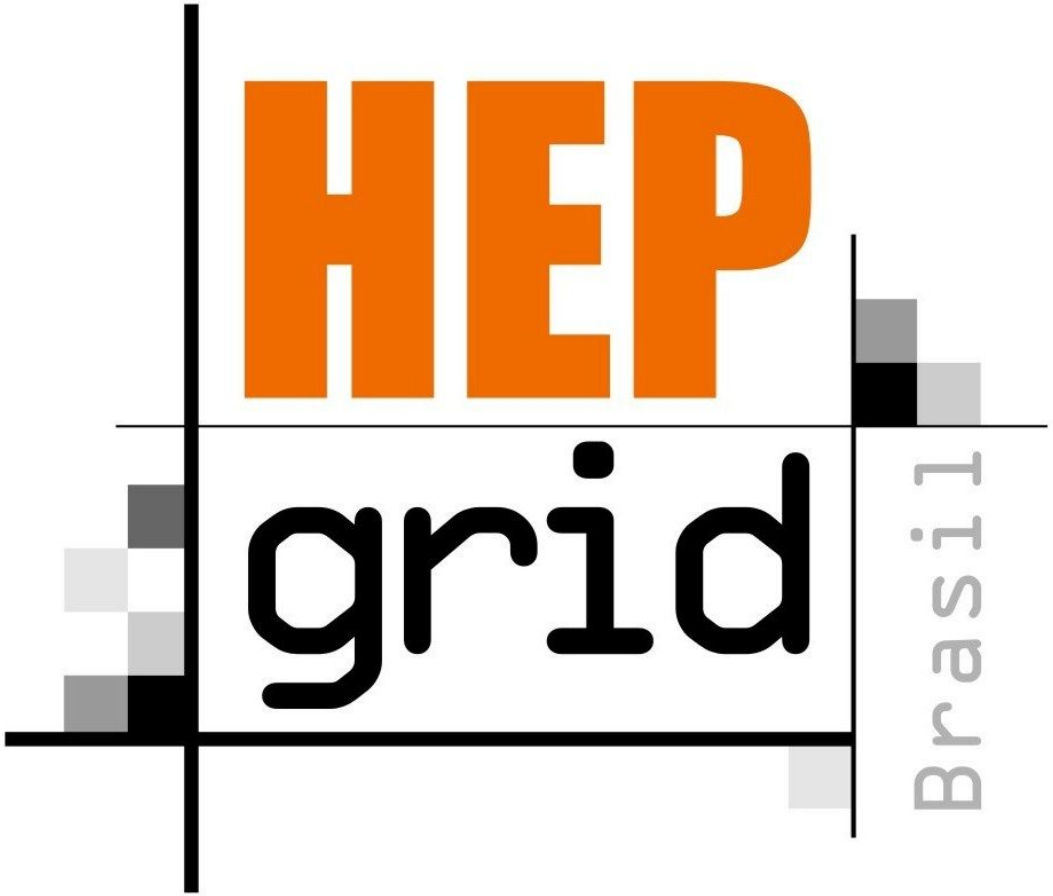
Introdução

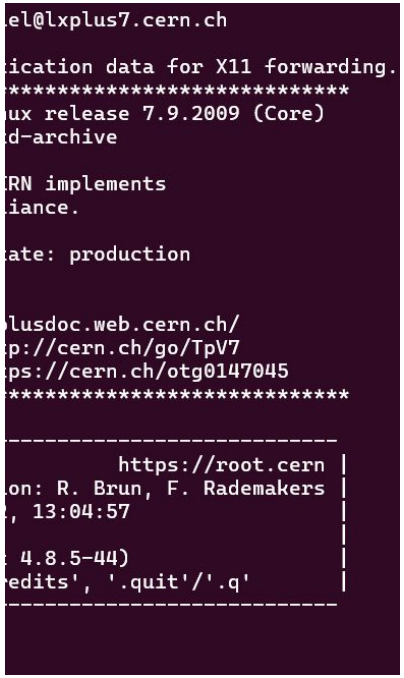
Alternativas:


```
(base) mthiel@mauricio-uerj:~$ ssh -Y mthiel@lxplus7.cern.ch
(mthiel@lxplus7.cern.ch) Password:
Warning: No xauth data; using fake authentication data for X11 forwarding.
* ****************************************************************
* Welcome to lxplus720.cern.ch, CentOS Linux release 7.9.2009 (Core)
* Archive of news is available in /etc/motd-archive
* Reminder: you have agreed to the CERN
*   computing rules, in particular OC5. CERN implements
*   the measures necessary to ensure compliance.
*   https://cern.ch/ComputingRules
* Puppet environment: production, Roger state: production
* Foreman hostgroup: lxplus/nodes/login
* Availability zone: cern-geneva-c
* LXPLUS Public Login Service - http://lxplusdoc.web.cern.ch/
* Please read LXPLUS Privacy Notice in http://cern.ch/go/TpV7
* 2024-06-27 - lxplus7 CC7 termination https://cern.ch/otg0147045
* ****************************************************************
[mthiel@lxplus720 ~]$ root -b
   ------------------------------------------------------------------
  | Welcome to ROOT 6.24/08                        https://root.cern |
  | (c) 1995-2021, The ROOT Team; conception: R. Brun, F. Rademakers |
  | Built for linuxx8664gcc on Sep 29 2022, 13:04:57                 |
  | From tags/v6-24-08@v6-24-08                                      |
  | With c++ (GCC) 4.8.5 20150623 (Red Hat 4.8.5-44)                 |
  | Try '.help', '.demo', '.license', '.credits', '.quit'/'.q'       |
   ------------------------------------------------------------------

root [0]
```

Introdução

Alternativas:

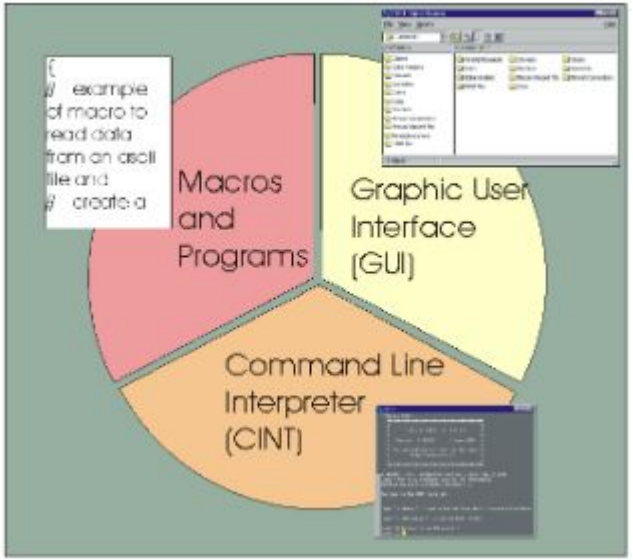




Interface com o ROOT

A interação com o ROOT pode ser feita de diferentes maneiras:

- Prompt de comando
- Macros
- Interface gráfica



Prompt do ROOT

```
[mthiel@lxplus720 ~]$ root
   ------------------------------------------------------------------
  | Welcome to ROOT 6.24/08                        https://root.cern |
  | (c) 1995-2021, The ROOT Team; conception: R. Brun, F. Rademakers |
  | Built for linuxx8664gcc on Sep 29 2022, 13:04:57                 |
  | From tags/v6-24-08@v6-24-08                                      |
  | With c++ (GCC) 4.8.5 20150623 (Red Hat 4.8.5-44)                 |
  | Try '.help', '.demo', '.license', '.credits', '.quit'/'.q'       |
   ------------------------------------------------------------------

root [0]
```

root -> inicializar o prompt do ROOT

root -l -> inicializar o ROOT sem mensagens de inicialização

root -b -> não permitir que o ROOT abra janelas

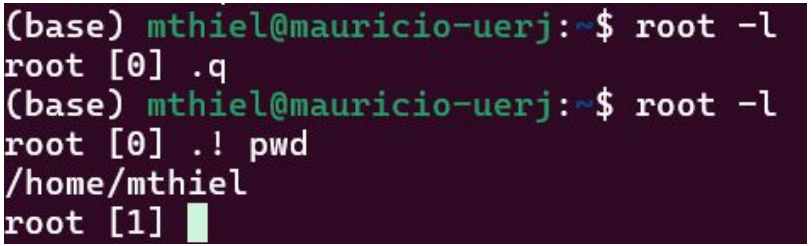
Prompt do ROOT

```
[mthiel@lxplus720 ~]$ root -h

usage: root [-b B] [-x X] [-e E] [-n N] [-t T] [-q Q] [-l L] [-a A]
            [-config CONFIG] [-memstat MEMSTAT] [-h HELP] [--version VERSION]
            [--notebook NOTEBOOK] [--web WEB] [--web=<browser> WEB=<BROWSER>]
            [dir] [file:data.root] [file1.C...fileN.C]

OPTIONS:
  -b                            Run in batch mode without graphics
  -x                            Exit on exceptions
  -e                            Execute the command passed between single quotes
  -n                            Do not execute logon and logoff macros as specified in .rootrc
  -t                            Enable thread-safety and implicit multi-threading (IMT)
  -q                            Exit after processing command line macro files
  -l                            Do not show the ROOT banner
  -a                            Show the ROOT splash screen
  -config                       print ./configure options
  -memstat                      run with memory usage monitoring
  -h, -?, --help                Show summary of options
  --version                     Show the ROOT version
  --notebook                    Execute ROOT notebook
  --web                         Display graphics in a default web browser
  --web=<browser>               Display graphics in specified web browser
  [dir]                         if dir is a valid directory cd to it before executing
  [file:data.root]              Open the ROOT file data.root
  [file1.C...fileN.C]           Execute the the ROOT macro file1.C ... fileN.C
[mthiel@lxplus720 ~]$
```

Prompt do ROOT

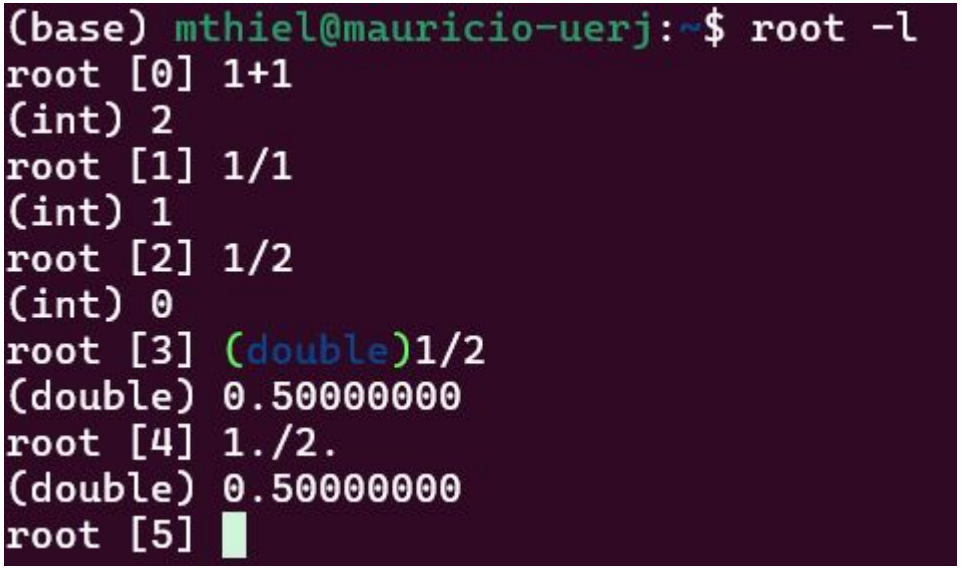
Principais comandos:

.q	Quit
.L macro.C	Load a macro file
.x macro.C	Load and execute macro file
.x macro.C++	Compile and execute
.! …	Executa comando fora do ambiente ROOT (ex: .! ls)
.help	Nos dá a lista de comandos



Prompt do ROOT

O prompt do ROOT é um interpretador de comandos em C++ já compilado



Prompt do ROOT

Aceita a criação de macros diretamente por linha de comando:

```
(base) mthiel@mauricio-uerj:~$ root -l
root [0] {
root (cont'ed, cancel with .@) [1]for(int i=0;i<10;i++){
root (cont'ed, cancel with .@) [2]cout << "i: " << i << endl;
root (cont'ed, cancel with .@) [3]}
root (cont'ed, cancel with .@) [4]}
i: 0
i: 1
i: 2
i: 3
i: 4
i: 5
i: 6
i: 7
i: 8
i: 9
root [5]
```

```
(base) mthiel@mauricio-uerj:~$ root -l
root [0] int dobro(int x){
root (cont'ed, cancel with .@) [1]return x*x;
root (cont'ed, cancel with .@) [2]}
root [3]
root [3]
root [3] dobro(2)
(int) 4
root [4] dobro(3)
(int) 9
root [5] dobro(100)
(int) 10000
root [6]
```

Macros no ROOT

- Em desenvolvimentos mais complexos, o uso do prompt do ROOT se torna inconveniente
- O uso de macros auxilia em desenvolvimento mais complexos

Macros no ROOT

- Em desenvolvimentos mais complexos, o uso do prompt do ROOT se torna inconveniente
- O uso de macros auxilia em desenvolvimento mais complexos

*simpleHistogram.C
~/

Open Save

```
1  void simpleHistogram() {
2      // Crie um histograma
3      TH1F *hist = new TH1F("hist", "Histograma de Números Aleatórios", 100, 0, 10);
4
5      // Preencha o histograma com números aleatórios
6      for (int i = 0; i < 10000; ++i) {
7          double randomNumber = gRandom->Uniform(0, 10);  // Gera um número aleatório entre 0 e 10
8          hist->Fill(randomNumber);
9      }
10
11     // Crie uma canvas para exibir o histograma
12     TCanvas *c1 = new TCanvas("c1", "Canvas com Histograma", 800, 600);
13     hist->Draw();
14
15     // Salve o histograma em um arquivo
16     c1->SaveAs("histograma.png");
17
18 }
```

C++ Tab Width: 8 Ln 17, Col 1 INS

Macros no ROOT

- Em desenvolvimentos mais complexos, o uso do prompt do ROOT se torna inconveniente
- O uso de macros auxilia em desenvolvimento mais complexos

.L macro.C	Load a macro file
.x macro.C	Load and execute macro file
.x macro.C++	Compile and execute

Open *simpleHistogram.C ~/ Save

```
1  void simpleHistogram() {
2      // Crie um histograma
3      TH1F *hist = new TH1F("hist", "Histograma de Números Aleatórios", 100, 0, 10);
4
5      // Preencha o histograma com números aleatórios
6      for (int i = 0; i < 10000; ++i) {
7          double randomNumber = gRandom->Uniform(0, 10);  // Gera um número aleatório entre 0 e 10
8          hist->Fill(randomNumber);
9      }
10
11     // Crie uma canvas para exibir o histograma
12     TCanvas *c1 = new TCanvas("c1", "Canvas com Histograma", 800, 600);
13     hist->Draw();
14
15     // Salve o histograma em um arquivo
16     c1->SaveAs("histograma.png");
17
18 }
```

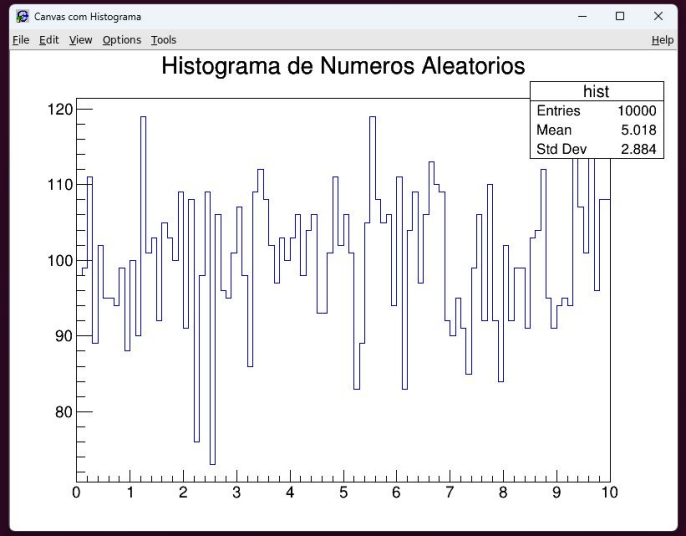
C++ Tab Width: 8 Ln 17, Col 1 INS

Macros no ROOT

- Em desenvolvimentos mais complexos, o uso do prompt do ROOT se torna inconveniente
- O uso de macros auxilia em desenvolvimento mais complexos

.L macro.C	Load a macro file
.x macro.C	Load and execute macro file
.x macro.C++	Compile and execute

```
(base) mthiel@mauricio-uerj:~$ root -l
root [0] .x simpleHistogram.C
Info in <TCanvas::Print>: file histograma.png has been created
root [1]
```



*simpleHistogram.C
~/

Save

```
simpleHistogram() {
// Crie um histograma
TH1F *hist = new TH1F("hist", "Histograma de Números Aleatórios", 100, 0, 10);

// Preencha o histograma com números aleatórios
for (int i = 0; i < 10000; ++i) {
    double randomNumber = gRandom->Uniform(0, 10);  // Gera um número aleatório entre 0 e 10
    hist->Fill(randomNumber);
}

// Crie uma canvas para exibir o histograma
TCanvas *c1 = new TCanvas("c1", "Canvas com Histograma", 800, 600);
hist->Draw();

// Salve o histograma em um arquivo
c1->SaveAs("histograma.png");
```

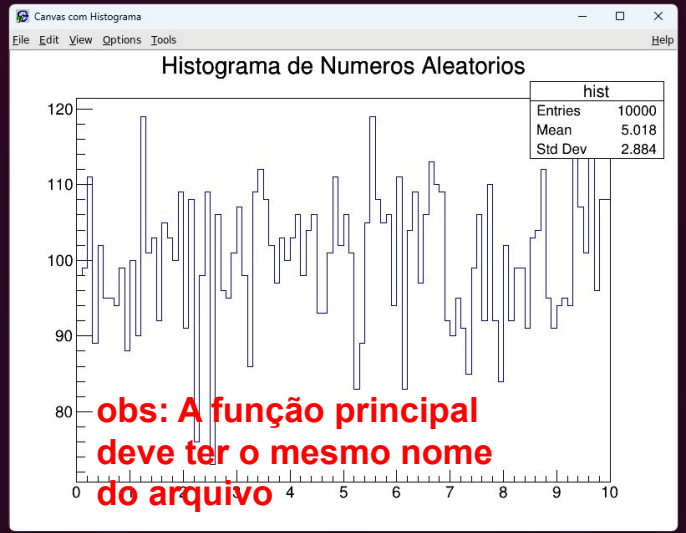
C++ Tab Width: 8 Ln 17, Col 1 INS

Macros no ROOT

- Em desenvolvimentos mais complexos, o uso do prompt do ROOT se torna inconveniente
- O uso de macros auxilia em desenvolvimento mais complexos

.L macro.C	Load a macro file
.x macro.C	Load and execute macro file
.x macro.C++	Compile and execute

```
(base) mthiel@mauricio-ueri: $ root -l
root [0] .x simpleHistogram.C
Info in <TCanvas::Print>: file histograma.png has been created
root [1]
```



Open *simpleHistogram.C ~/ Save

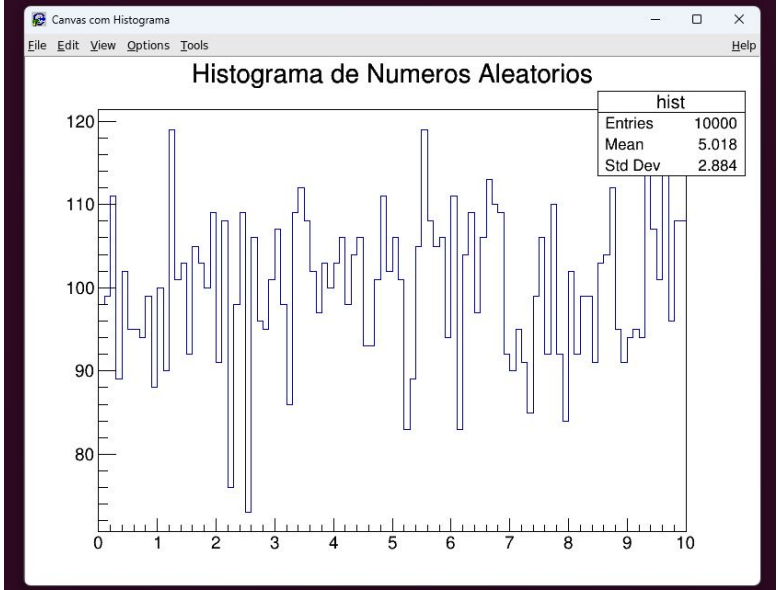
```
void simpleHistogram() {
    // Crie um histograma
    TH1F *hist = new TH1F("hist", "Histograma de Números Aleatórios", 100, 0, 10);

    // Preencha o histograma com números aleatórios
    for (int i = 0; i < 10000; ++i) {
        double randomNumber = gRandom->Uniform(0, 10);  // Gera um número aleatório entre 0 e
        hist->Fill(randomNumber);
    }

    // Crie uma canvas para exibir o histograma
    TCanvas *c1 = new TCanvas("c1", "Canvas com Histograma", 800, 600);
    hist->Draw();

    // Salve o histograma em um arquivo
    c1->SaveAs("histograma.png");

}
```

C++ Tab Width: 8 Ln 17, Col 1

Macros no ROOT

- Em desenvolvimentos mais complexos, o uso do prompt do ROOT se torna inconveniente
- O uso de macros auxilia em desenvolvimento mais complexos

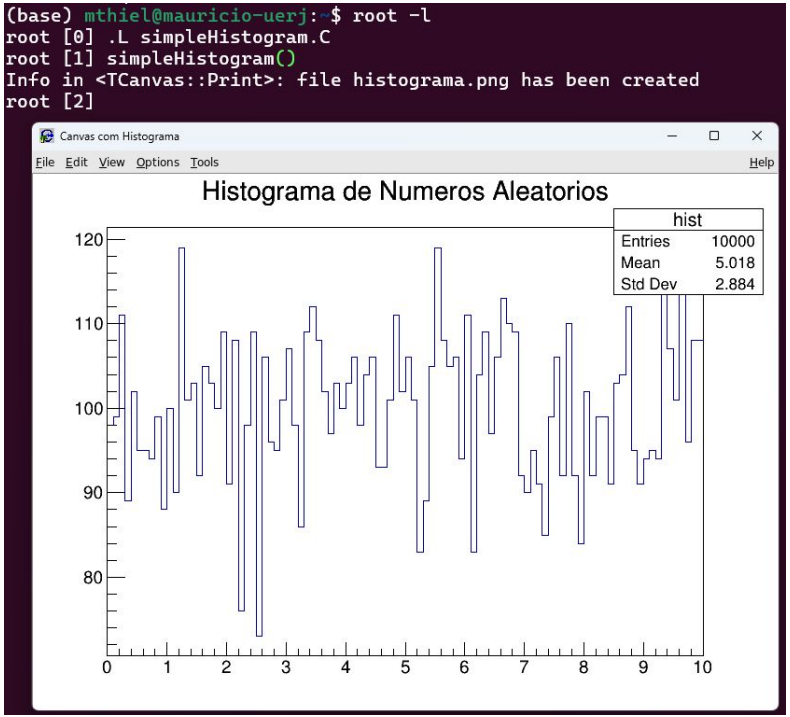
```
(base) mthiel@mauricio-uerj:~$ root -l simpleHistogram.C
root [0]
Processing simpleHistogram.C...
Info in <TCanvas::Print>: file histograma.png has been created
root [1]
```



obs: A função principal deve ter o mesmo nome do arquivo

Macros no ROOT

- Em desenvolvimentos mais complexos, o uso do prompt do ROOT se torna inconveniente
- O uso de macros auxilia em desenvolvimento mais complexos



obs: A função principal NÃO precisa ter o mesmo nome do arquivo

Macros no ROOT

- Chamando mais funções

simpleHistogram.C
~/

```
1  void createHistogram(int N) {
2      // Crie um histograma
3      TH1F *hist = new TH1F("hist", "Histograma de Numeros Aleatorios", 100, 0, 10);
4
5      // Preencha o histograma com números aleatórios
6      for (int i = 0; i < N; ++i) {
7          double randomNumber = gRandom->Uniform(0, 10);  // Gera um número aleatório entre 0 e 10
8          hist->Fill(randomNumber);
9      }
10
11     // Crie uma canvas para exibir o histograma
12     TCanvas *c1 = new TCanvas("c1", "Canvas com Histograma", 800, 600);
13     hist->Draw();
14
15     // Salve o histograma em um arquivo
16     c1->SaveAs("histograma.png");
17
18 }
19
20 void simpleHistogram() {
21     createHistogram(5);
22 }
```

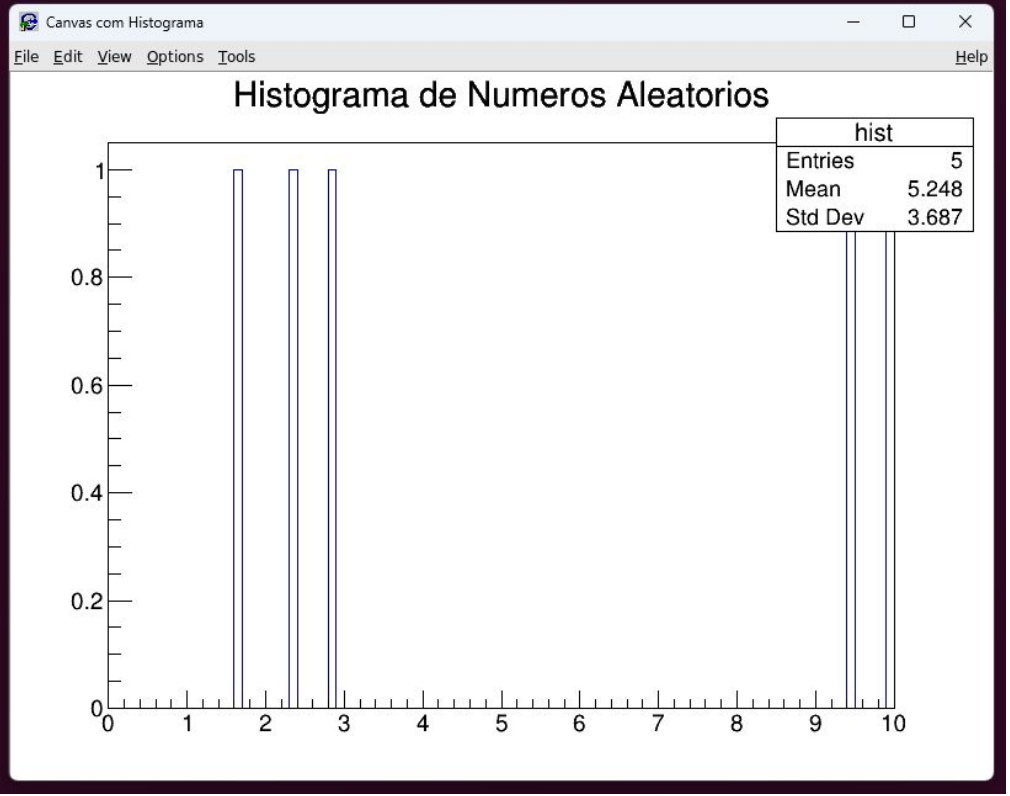
Macros no ROOT

- Chamando mais funções

simpleHistogram.C
~/

```
1  void createHistogram(int N) {
2      // Crie um histograma
3      TH1F *hist = new TH1F("hist", "Histograma de Numeros Ale
4
5      // Preencha o histograma com números aleatórios
6      for (int i = 0; i < N; ++i) {
7          double randomNumber = gRandom->Uniform(0, 10);  // G
8          hist->Fill(randomNumber);
9      }
10
11     // Crie uma canvas para exibir o histograma
12     TCanvas *c1 = new TCanvas("c1", "Canvas com Histograma",
13     hist->Draw();
14
15     // Salve o histograma em um arquivo
16     c1->SaveAs("histograma.png");
17
18 }
19
20 void simpleHistogram() {
21     createHistogram(5);
22 }
```

```
(base) mthiel@mauricio-uerj:~$ root -l simpleHistogram.C
root [0]
Processing simpleHistogram.C...
Info in <TCanvas::Print>: file histograma.png has been created
root [1]
```



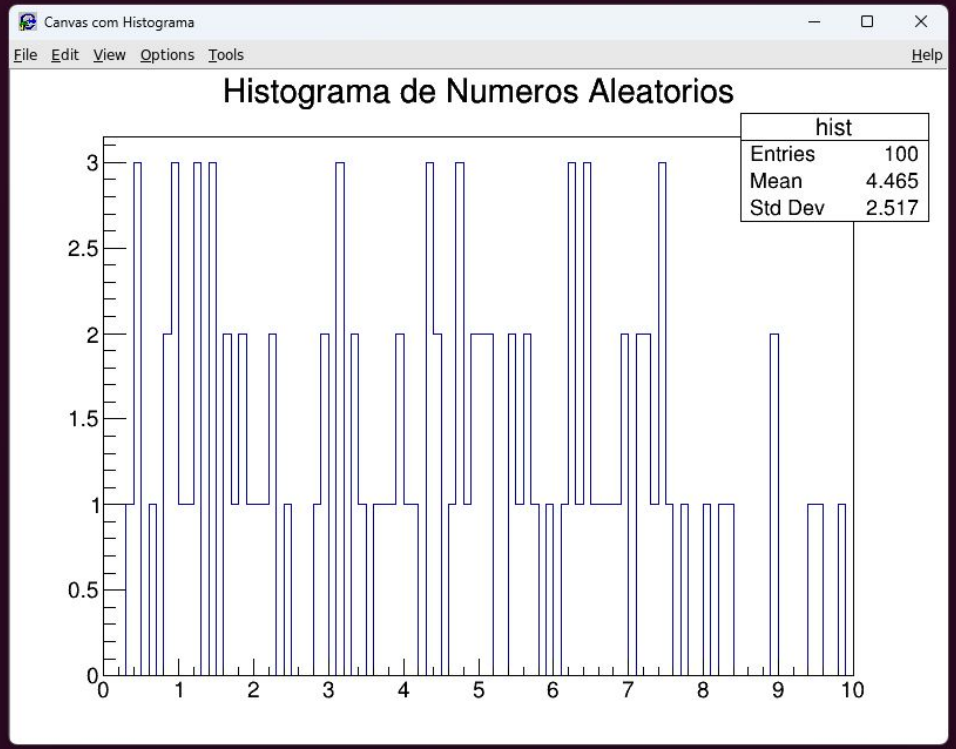
Macros no ROOT

- Chamando mais funções

Open ▾ simpleHistogra ~/

```
1  void createHistogram(int N) {
2      // Crie um histograma
3      TH1F *hist = new TH1F("hist", "Histograma de Num
4
5      // Preencha o histograma com números aleatórios
6      for (int i = 0; i < N; ++i) {
7          double randomNumber = gRandom->Uniform(0, 10
8          hist->Fill(randomNumber);
9      }
10
11     // Crie uma canvas para exibir o histograma
12     TCanvas *c1 = new TCanvas("c1", "Canvas com Hist
13     hist->Draw();
14
15     // Salve o histograma em um arquivo
16     c1->SaveAs("histograma.png");
17
18 }
19
20 void simpleHistogram() {
21     createHistogram(5);
22 }
```

```
(base) mthiel@mauricio-uerj:~$ root -l simpleHistogram.C
root [0]
Processing simpleHistogram.C...
Info in <TCanvas::Print>: file histograma.png has been created
root [1] cre
createHistogram
cregex_iterator
cregex_token_iterator
root [1] createHistogram(100)
Warning in <TROOT::Append>: Replacing existing TH1: hist (Potential memory leak).
Warning in <TCanvas::Constructor>: Deleting canvas with same name: c1
Info in <TCanvas::Print>: file histograma.png has been created
root [2]
```



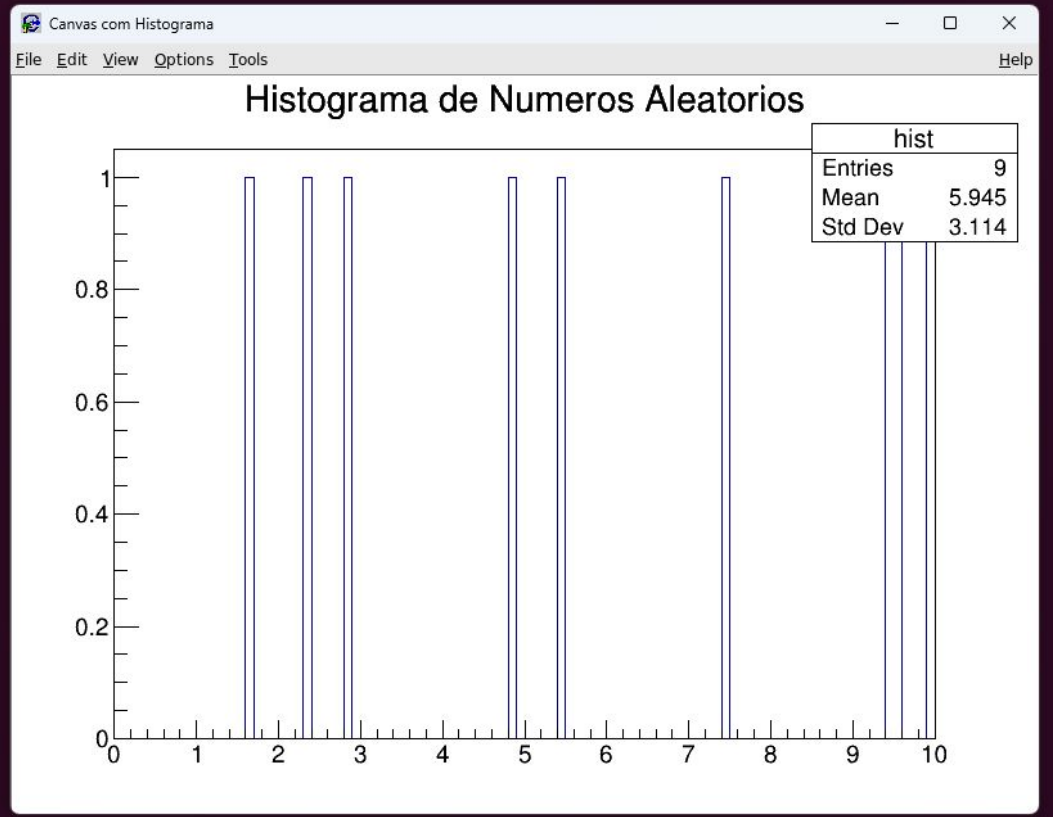
Macros no ROOT

- Chamando mais funções

simpleHistogram.C
~/

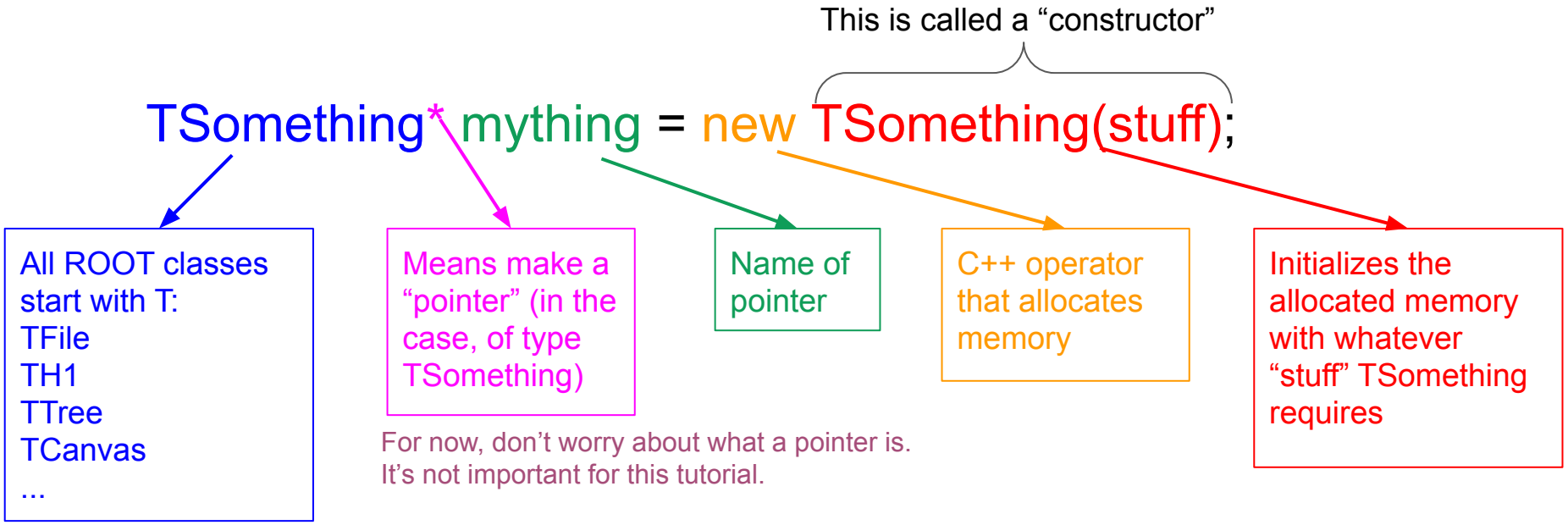
```
1  void createHistogram(int N) {
2      // Crie um histograma
3      TH1F *hist = new TH1F("hist", "Histograma de Numeros A
4
5      // Preencha o histograma com números aleatórios
6      for (int i = 0; i < N; ++i) {
7          double randomNumber = gRandom->Uniform(0, 10);  //
8          hist->Fill(randomNumber);
9      }
10
11     // Crie uma canvas para exibir o histograma
12     TCanvas *c1 = new TCanvas("c1", "Canvas com Histograma
13     hist->Draw();
14
15     // Salve o histograma em um arquivo
16     c1->SaveAs("histograma.png");
17
18 }
19
20 void simpleHistogram() {
21     createHistogram(5);
22 }
```

```
(base) mthiel@mauricio-uerj:~$ root -l
root [0] .L si
signal_fit/
simpleHistogram.C
root [0] .L simpleHistogram.C
root [1] createHistogram(9)
Info in <TCanvas::Print>: file histograma.png has been created
root [2]
```



Sintaxe do ROOT

Muitos dos comandos que vamos utilizar terão esta forma geral:



Note: In C++, if you you allocate memory using the “new” operator, you must later use “delete mything” to release the memory… otherwise your code will have a memory leak.
We will not worry about that today, but keep it in mind for your future code-writing

Sintax

C++

Muitos d

TS

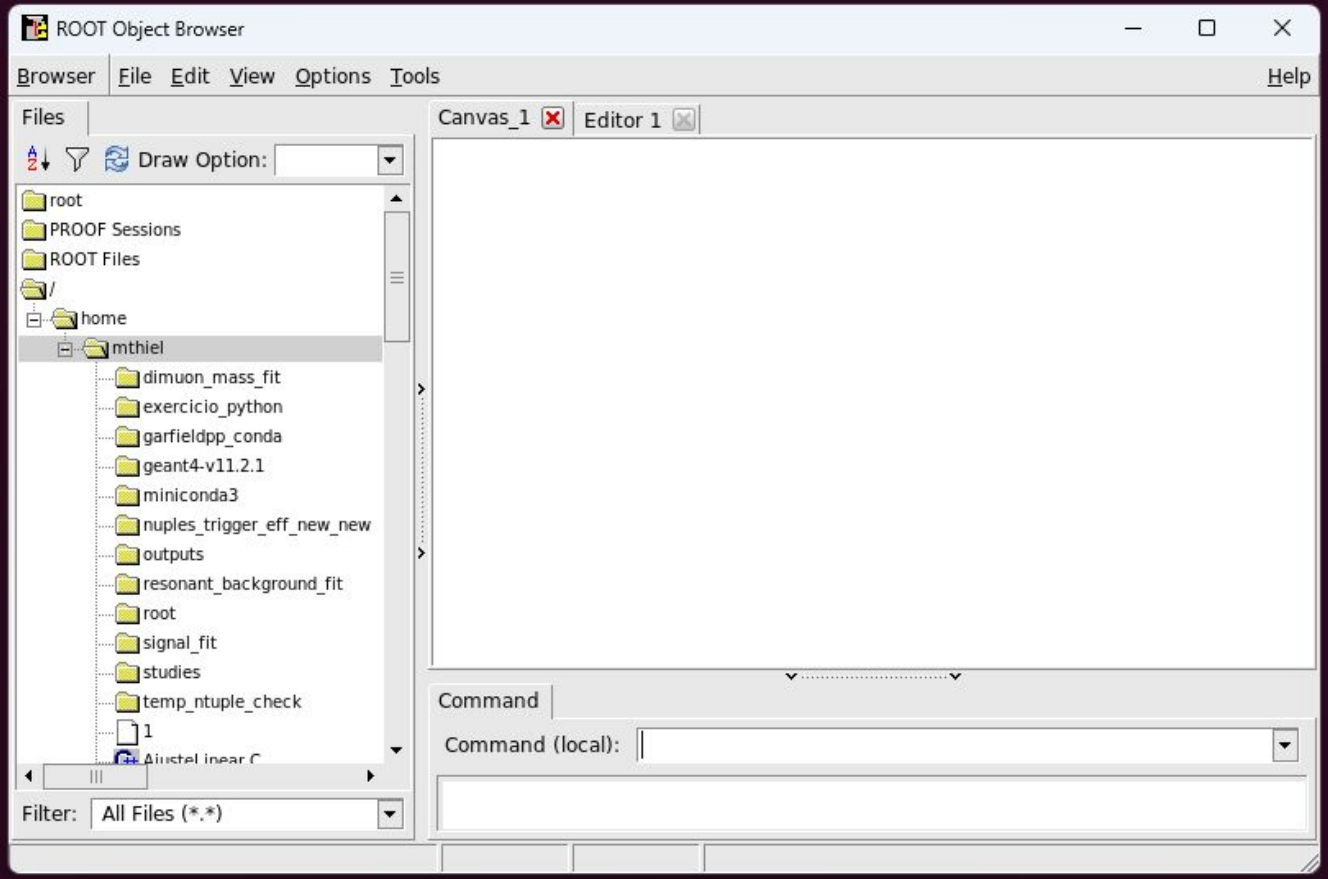
All ROOT cla
start with T:
TFile
TH1
TTree
TCanvas
...

es the
ed memory
natever
TSomething
s

Note: In
memory.
We will n

elease the

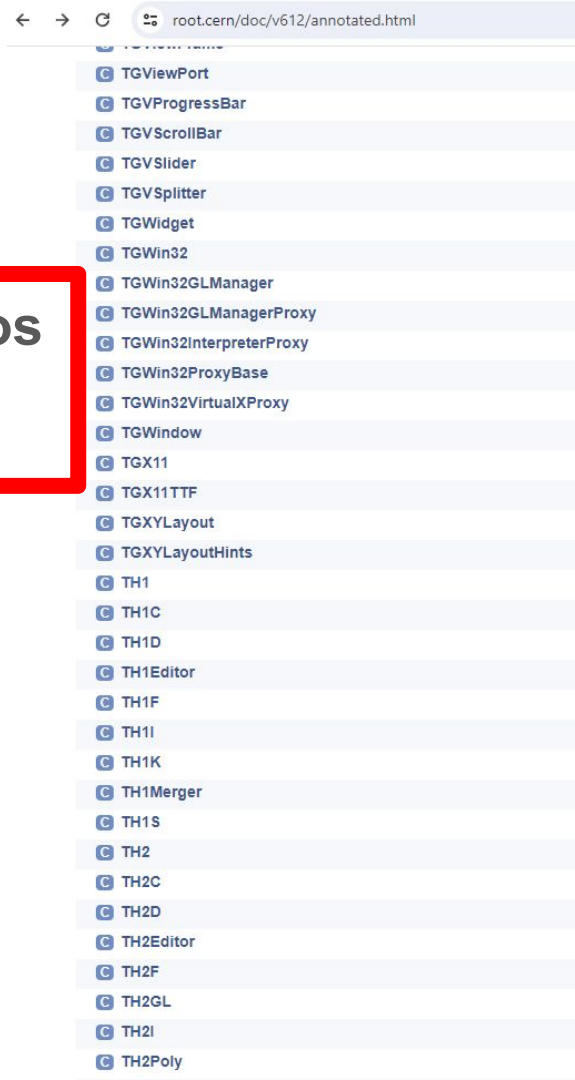
```
(base) mthiel@mauricio-uerj:~$ root -l
root [0] TBrowser* j1 = new TBrowser()
(TBrowser *) 0x5624d8075440
root [1]
```



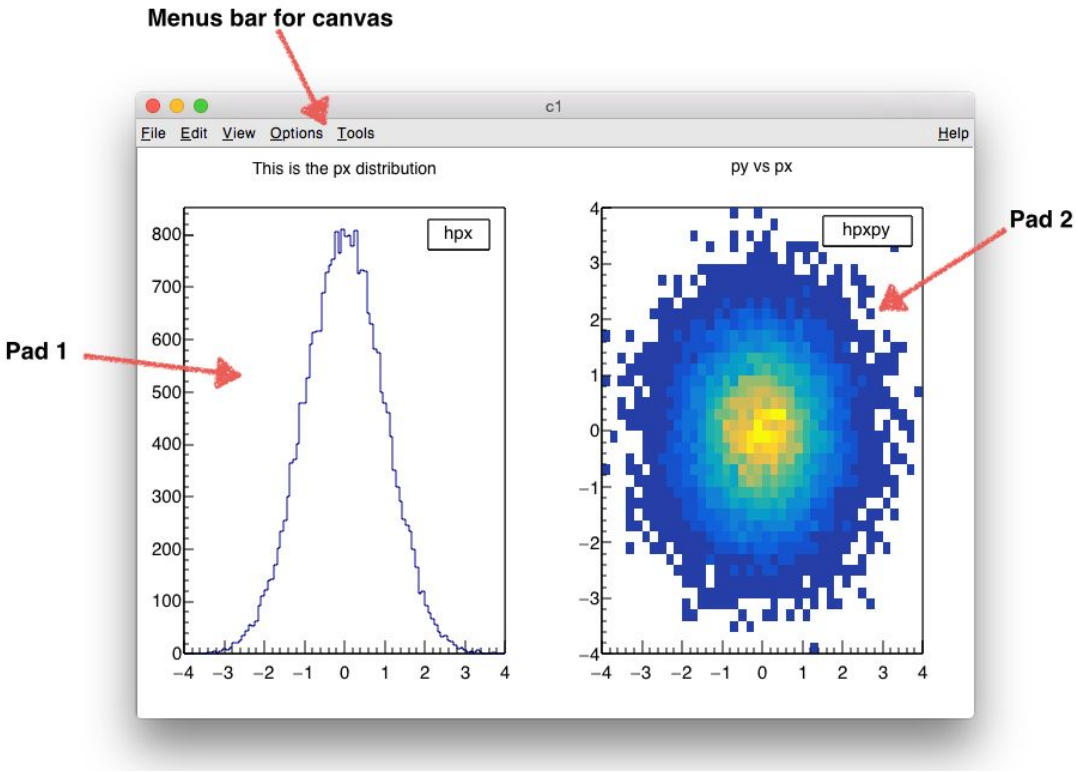
Classes do ROOT

Uma classe é uma estrutura que encapsula objetos e funções que operam sobre esses objetos

- Por convenção no ROOT, classes começam com T
- Com essas classes podemos criar janelas, histogramas, gráficos, legenda para os gráficos, manipular os elementos dos gráficos/histogramas, fazer fit, gerar números aleatórios e uma infinidade ações
- Lista de classes do ROOT:
  - https://root.cern/doc/v612/annotated.html

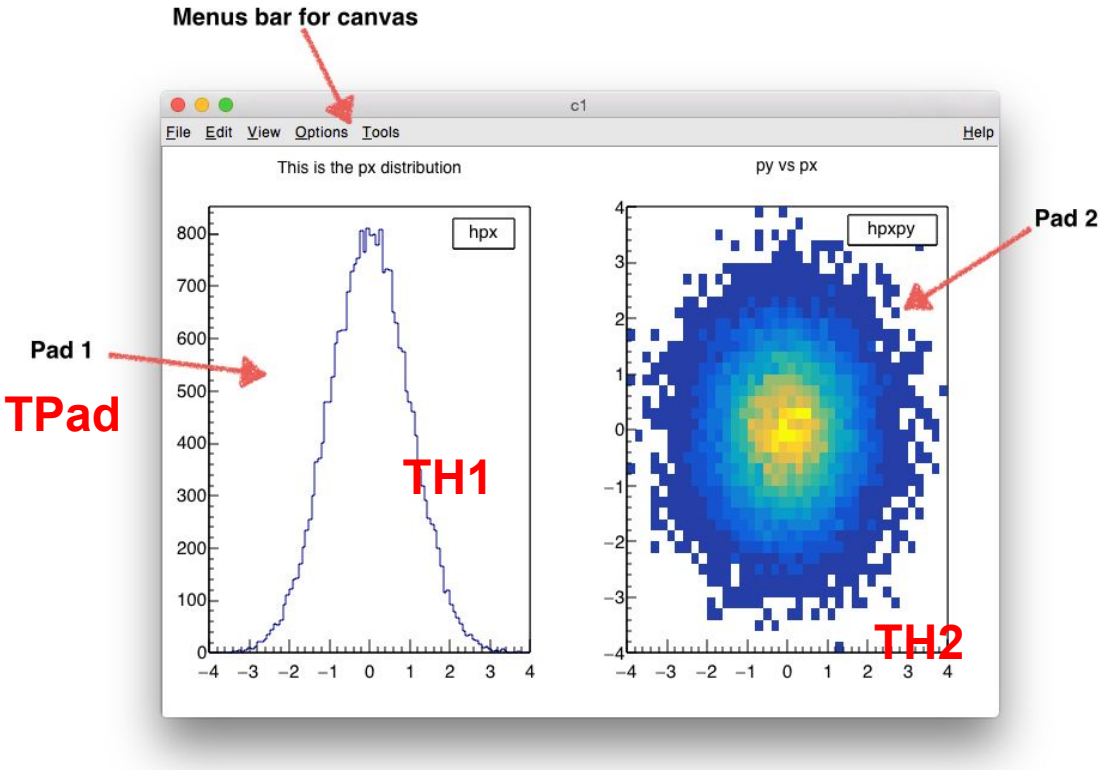


Classes do ROOT: Canvas - TCanvas



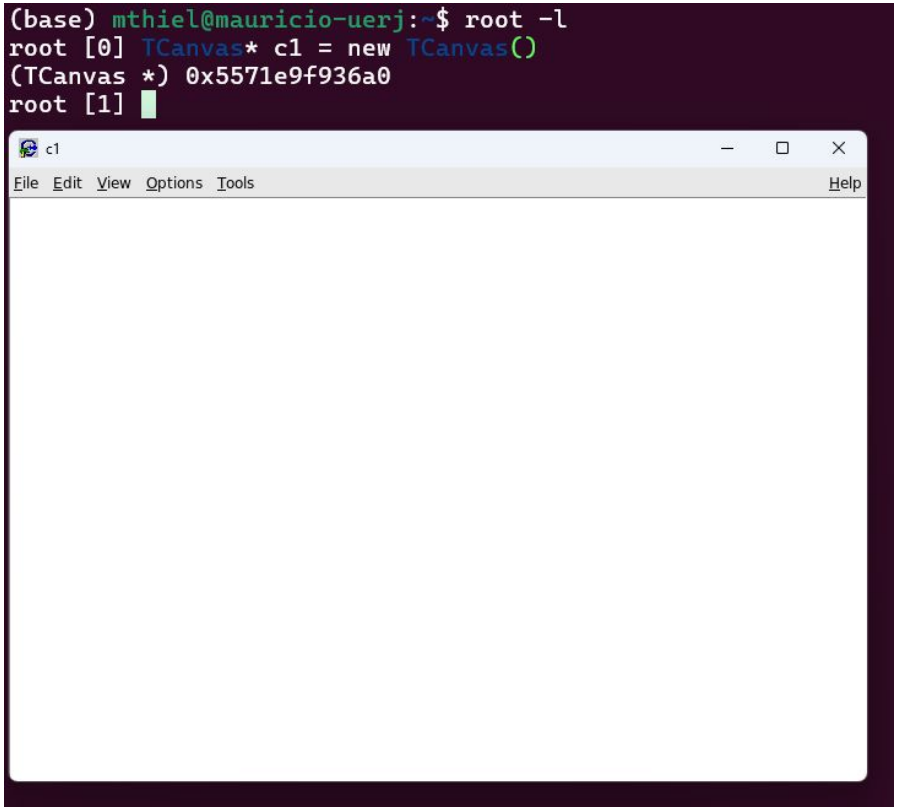
A tela onde será “pintado” os objetos como histograma, funções, etc

Classes do ROOT: Canvas - TCanvas

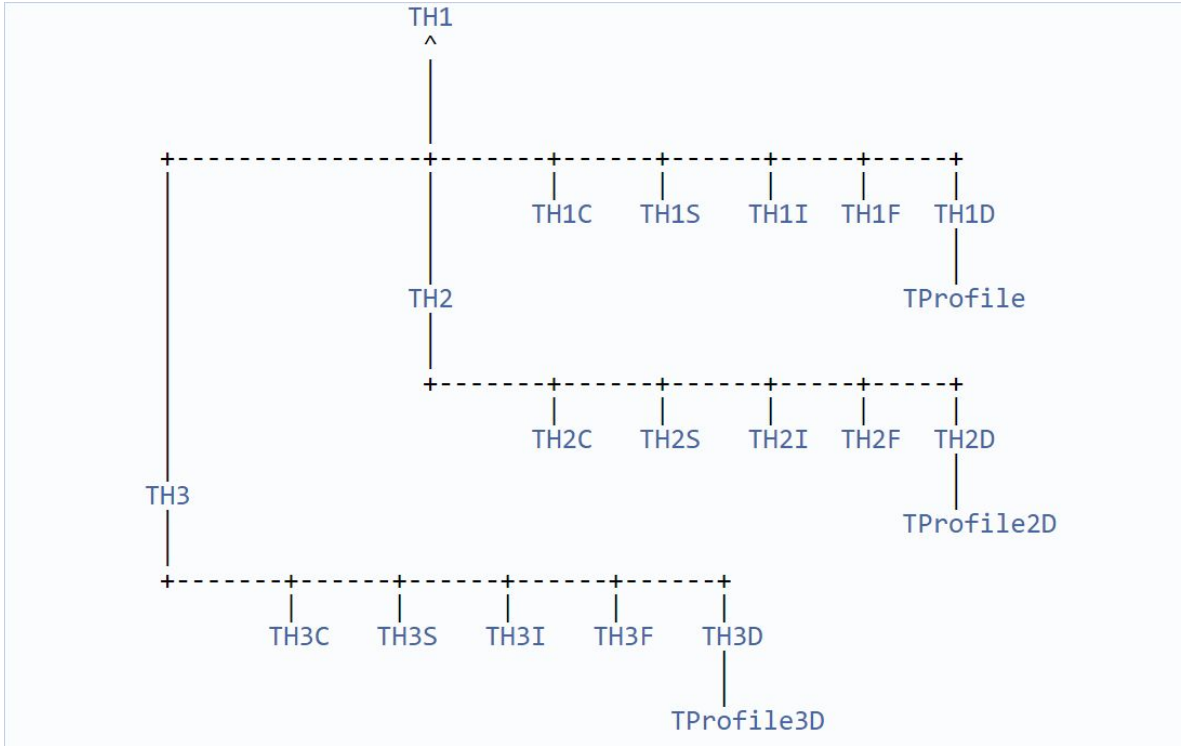


A tela onde será “pintado” os objetos como histograma, funções, etc

Classes do ROOT: Canvas - TCanvas

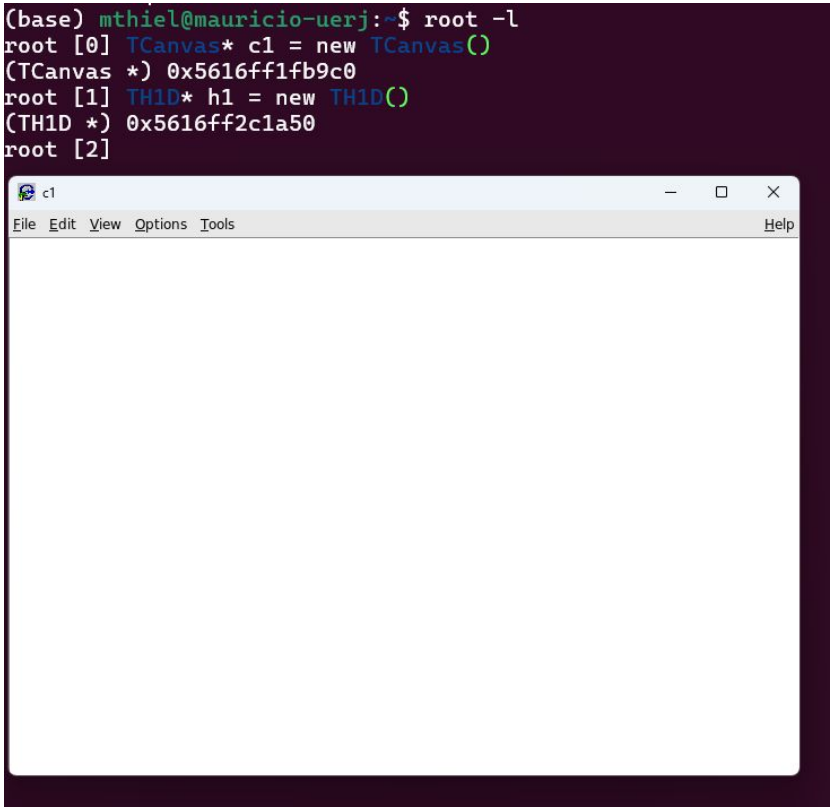


Classes do ROOT: Histogramas - TH1



- Classe TH1
  - TH1F, TH1D
  - TH2F, TH2D
  - TH3F, TH3D
- Onde serão construídos nossos histogramas

Classes do ROOT: Histogramas - TH1



Classes do ROOT: Histogramas - TH1

https://root.cern/doc/v612/classTH1.html

função da classe TH1

Draw()

`void TH1::Draw ( Option_t * option = "" )` *virtual*

Draw this histogram with options.

Histograms are drawn via the **THistPainter** class. Each histogram has a pointer to its own painter (to be usable in a multithreaded program). The same histogram can be drawn with different options in different pads. When an histogram drawn in a pad is deleted, the histogram is automatically removed from the pad or pads where it was drawn. If an histogram is drawn in a pad, then filled again, the new status of the histogram will be automatically shown in the pad next time the pad is updated. One does not need to redraw the histogram. To draw the current version of an histogram in a pad, one can use `h->DrawCopy();` This makes a clone of the histogram. Once the clone is drawn, the original histogram may be modified or deleted without affecting the aspect of the clone. By default, **TH1::Draw** clears the current pad.

One can use **TH1::SetMaximum** and **TH1::SetMinimum** to force a particular value for the maximum or the minimum scale on the plot.

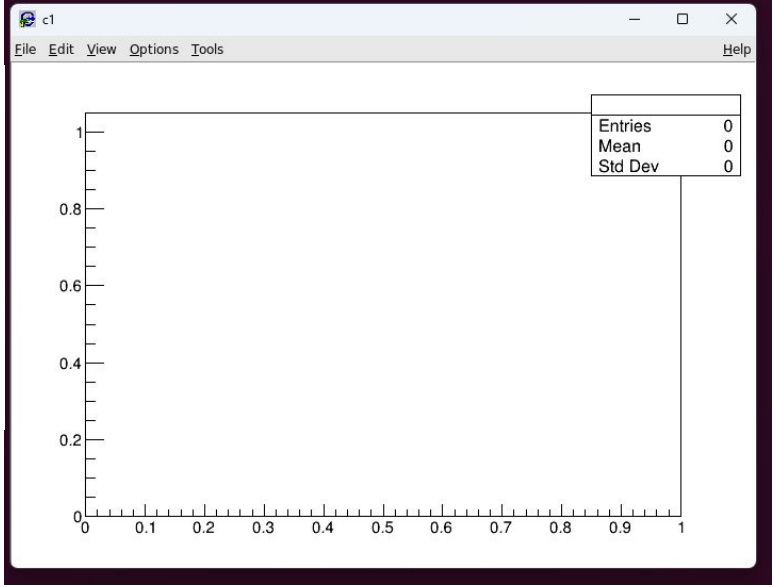
TH1::UseCurrentStyle can be used to change all histogram graphics attributes to correspond to the current selected style. This function must be called for each histogram. In case one reads and draws many histograms from a file, one can force the histograms to inherit automatically the current graphics style by calling before gROOT->ForceStyle();

See the **THistPainter** class for a description of all the drawing options.

Reimplemented from **TObject**.

Definition at line **2969** of file **TH1.cxx**.

```
(base) mthiel@mauricio-uerj:~$ root -l
root [0] TH1D* h1 = new TH1D()
(TH1D *) 0x556144416e00
root [1] .q
(base) mthiel@mauricio-uerj:~$ root -l
root [0] TCanvas* c1 = new TCanvas()
(TCanvas *) 0x5616ff1fb9c0
root [1] TH1D* h1 = new TH1D()
(TH1D *) 0x5616ff2c1a50
root [2] h1->Draw()
root [3]
```



Classes do ROOT: Histogramas - TH1

https://root.cern/doc/v612/classTH1D.html

Public Member Functions

TH1D ()
Constructor. More...

TH1D (const char *name, const char *title, **Int_t** nbinsx, **Double_t** xlow, **Double_t** xup)
Create a 1-Dim histogram with fix bins of type double (see **TH1::TH1** for explanation of parameters) More...

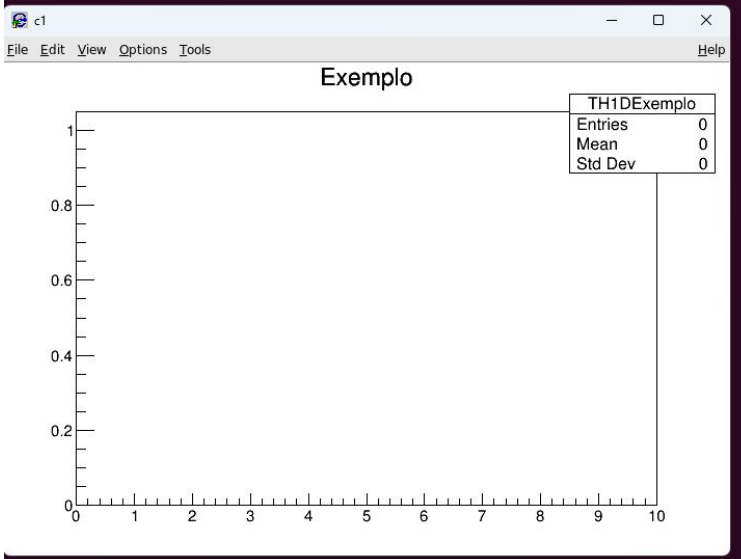
TH1D (const char *name, const char *title, **Int_t** nbinsx, const **Float_t** *xbins)
Create a 1-Dim histogram with variable bins of type double (see **TH1::TH1** for explanation of parameters) More...

TH1D (const char *name, const char *title, **Int_t** nbinsx, const **Double_t** *xbins)
Create a 1-Dim histogram with variable bins of type double (see **TH1::TH1** for explanation of parameters) More...

TH1D (const **TVectorD** &v)
Create a histogram from a TVectorD by default the histogram name is "TVectorD" and title = "". More...

TH1D (const **TH1D** &h1d)
Constructor. More...

```
(base) mthiel@mauricio-uerj:~$ root -l
root [0] TCanvas* c1 = new TCanvas()
(TCanvas *) 0x55afc1931700
root [1] TH1D *h1 = new TH1D("TH1DExemplo", "Exemplo", 10, 0, 10)
(TH1D *) 0x55afc1aab2a0
root [2] h1->Draw()
root [3]
```

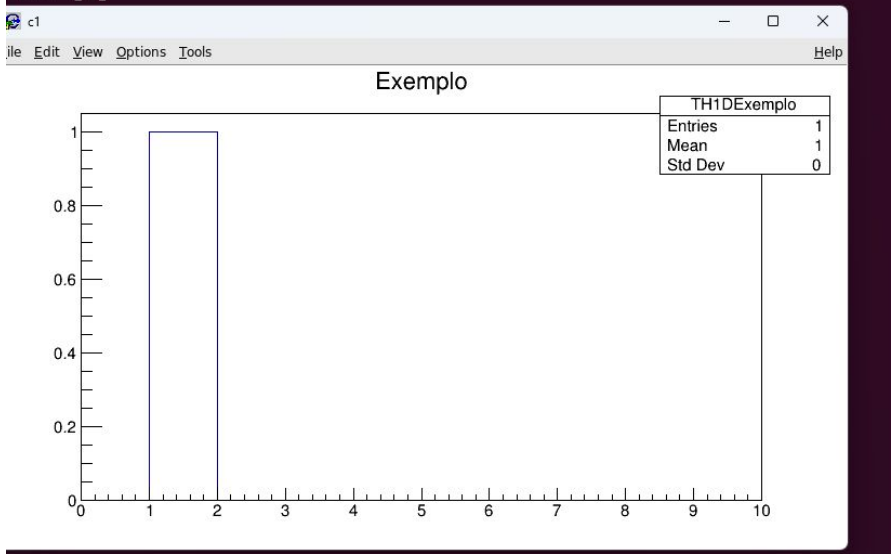


Classes do ROOT: Histogramas - TH1

https://root.cern/doc/v612/classTH1.html

virtual Int_t	Fill (Double_t x) Increment bin with abscissa X by 1. More...
virtual Int_t	Fill (Double_t x, Double_t w) Increment bin with abscissa X with a weight w. More...
virtual Int_t	Fill (const char *name, Double_t w) Increment bin with namex with a weight w. More...

```
(base) mthiel@mauricio-uerj:~$ root -l
root [0] TCanvas* c1 = new TCanvas()
(TCanvas *) 0x55afc1931700
root [1] TH1D *h1 = new TH1D("TH1DExemplo", "Exemplo", 10, 0, 10)
(TH1D *) 0x55afc1aab2a0
root [2] h1->Draw()
root [3] h1->Fill(1)
(int) 2
root [4] h1->Draw()
root [5]
```



Classes do ROOT: Histogramas - TH1

https://root.cern/doc/v612/classTH1.html

◆ GetXaxis() [1/2]

TAxis* TH1::GetXaxis ( )

Get the behaviour adopted by the object about the statoverflows. See EStatOverflows for more information.

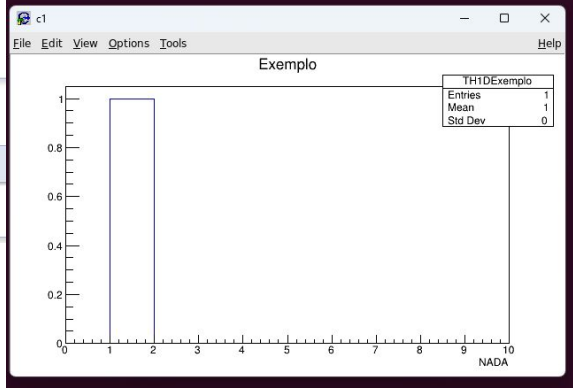
Definition at line 315 of file TH1.h.

◆ GetXaxis() [2/2]

const TAxis* TH1::GetXaxis ( ) const

Definition at line 318 of file TH1.h.

```
(base) mthiel@mauricio-uerj:~$ root -l
root [0] TCanvas* c1 = new TCanvas()
(TCanvas *) 0x55afc1931700
root [1] TH1D *h1 = new TH1D("TH1DExemplo", "Exemplo", 10, 0, 10)
(TH1D *) 0x55afc1aab2a0
root [2] h1->Draw()
root [3] h1->Fill(1)
(int) 2
root [4] h1->Draw()
root [5] h1->GetXaxis()->SetTitle("NADA")
root [6] h1->Draw()
root [7]
```



Classes do ROOT: Histogramas - TH1

https://root.cern/doc/v612/classTH1.html

GetXaxis() [1/2]

TAxis* TH1::GetXaxis ( )

Get the behaviour adopted by the object about the statoverflows. See EStatOverflows for more information.

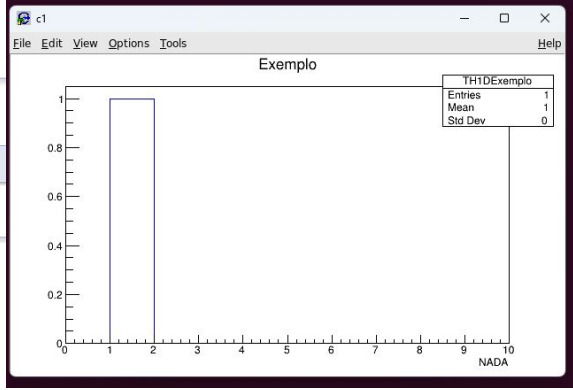
Definition at line 315 of file TH1.h.

GetXaxis() [2/2]

cons TAxis* TH1::GetXaxis ( ) const

Definition at line 318 of file TH1.h.

```
(base) mthiel@mauricio-uerj:~$ root -l
root [0] TCanvas* c1 = new TCanvas()
(TCanvas *) 0x55afc1931700
root [1] TH1D *h1 = new TH1D("TH1DExemplo", "Exemplo", 10, 0, 10)
(TH1D *) 0x55afc1aab2a0
root [2] h1->Draw()
root [3] h1->Fill(1)
(int) 2
root [4] h1->Draw()
root [5] h1->GetXaxis()->SetTitle("NADA")
root [6] h1->Draw()
root [7]
```



função GetXaxis() retorna um objeto da classe TAxis: https://root.cern/doc/v612/classTAxis.html

Classes do ROOT: Histogramas - TH1

https://root.cern/doc/v612/classTH1.html

◆ GetXaxis() [1/2]

TAxis* TH1::GetXaxis ( )

Get the behaviour adopted by the object about the statoverflows. See EStatOverflows for more information.

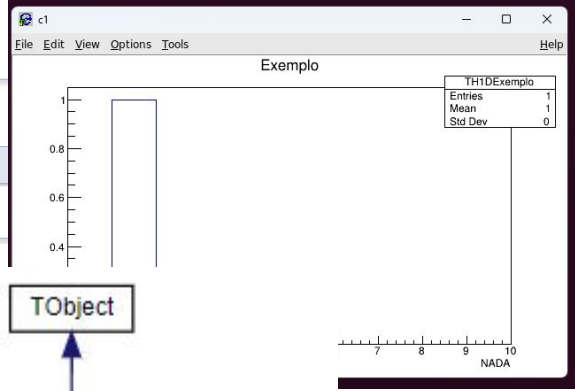
Definition at line 315 of file TH1.h.

◆ GetXaxis() [2/2]

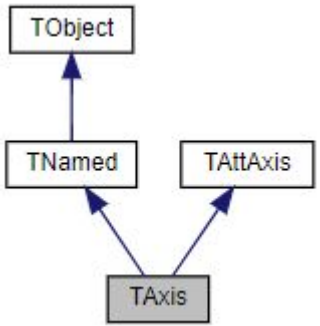
cons TAxis* TH1::GetXaxis ( ) const

Definition at line 318 of file TH1.h.

```
(base) mthiel@mauricio-uerj:~$ root -l
root [0] TCanvas* c1 = new TCanvas()
(TCanvas *) 0x55afc1931700
root [1] TH1D *h1 = new TH1D("TH1DExemplo", "Exemplo", 10, 0, 10)
(TH1D *) 0x55afc1aab2a0
root [2] h1->Draw()
root [3] h1->Fill(1)
(int) 2
root [4] h1->Draw()
root [5] h1->GetXaxis()->SetTitle("NADA")
root [6] h1->Draw()
root [7]
```



função GetXaxis() retorna um objeto da classe TAxis: https://root.cern/doc/v612/classTAxis.html



Classes do ROOT: Histogramas - TH1

https://root.cern/doc/v612/classTNamed.html

◆ SetTitle()

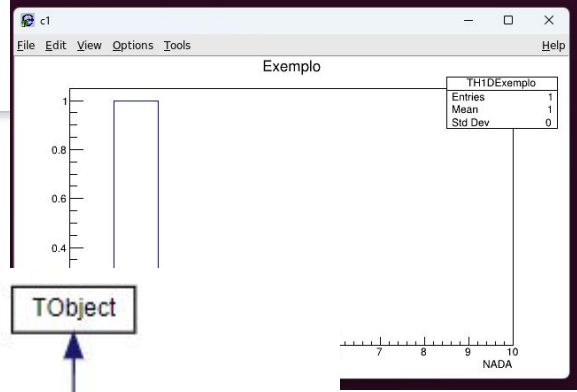
void TNamed::SetTitle (const char * title = "")

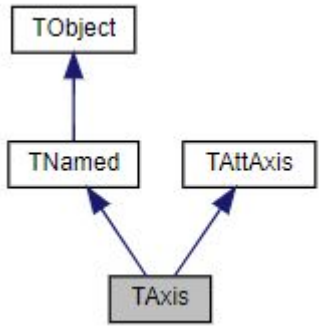
Set the title of the TNamed.

Reimplemented in TF1, TH1, TGraph, THnBase, TGraph2D, TEfficiency, TParallelCoordVar, TASImage, RooPlot, TSystemDirectory, and TSystemFile.

Definition at line 164 of file TNamed.cxx.

```
(base) mthiel@mauricio-uerj:~$ root -l
root [0] TCanvas* c1 = new TCanvas()
(TCanvas *) 0x55afc1931700
root [1] TH1D *h1 = new TH1D("TH1DExemplo", "Exemplo", 10, 0, 10)
(TH1D *) 0x55afc1aab2a0
root [2] h1->Draw()
root [3] h1->Fill(1)
(int) 2
root [4] h1->Draw()
root [5] h1->GetXaxis()->SetTitle("NADA")
root [6] h1->Draw()
root [7]
```

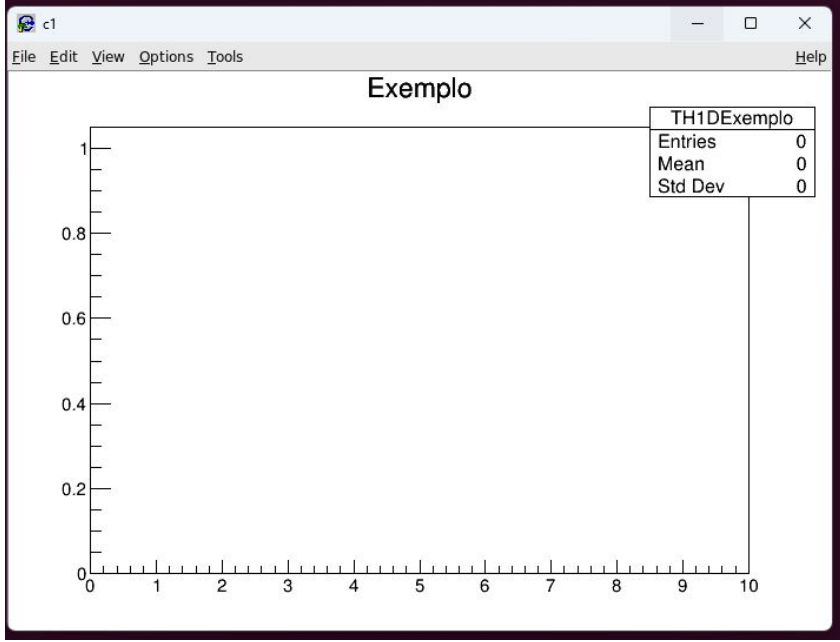




Classes do ROOT: Histogramas - TH1

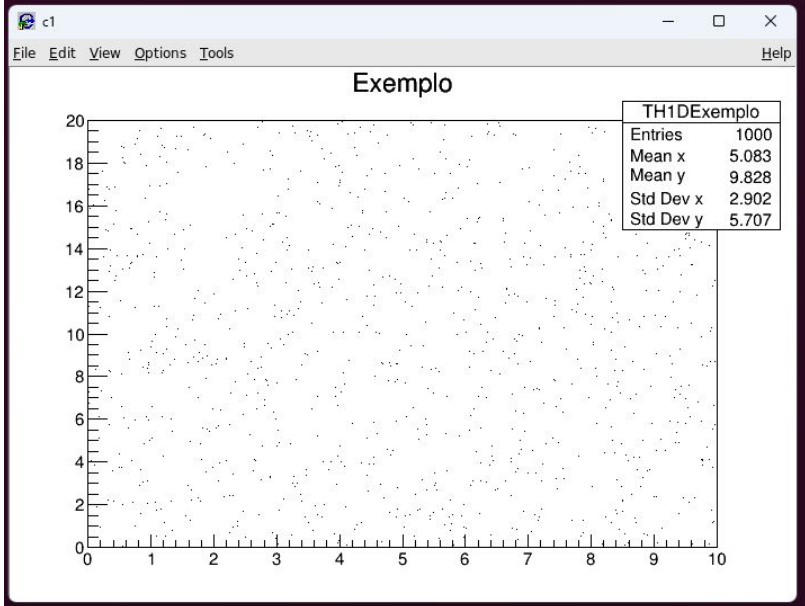
- Criando histograma sem criar o TCanvas:
  - ele é gerado automaticamente

```
(base) mthiel@mauricio-uerj:~$ root -l
root [0] TH1D *h1 = new TH1D("TH1DExemplo", "Exemplo", 10, 0, 10)
(TH1D *) 0x5557020d1820
root [1] h1->Draw()
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
root [2]
```



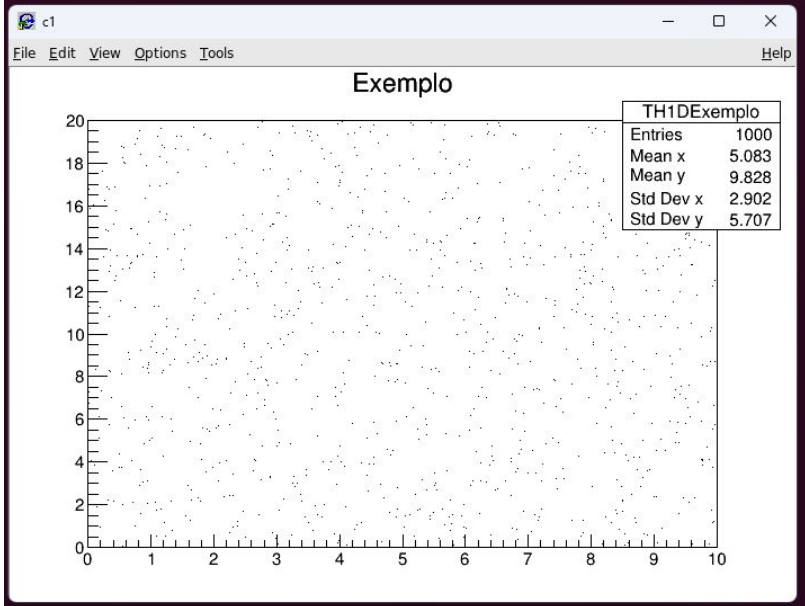
Classes do ROOT: Histogramas - TH2

```
(base) mthiel@mauricio-uerj: $ root -l
root [0] TH2D *h2 = new TH2D("TH1DExemplo", "Exemplo", 10, 0, 10, 20, 0, 20)
(TH2D *) 0x55a64c304ab0
root [1] for (Int_t i = 0; i < 1000; ++i) {Double_t x = gRandom->Uniform(0, 10);  Double_t y = gRandom->Uniform(0, 20);h2->Fill(x, y);}
root [2] h2->Draw()
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
root [3]
```



Classes do ROOT: Histogramas - TH2

```
(base) mthiel@mauricio-uerj: $ root -l
root [0] TH2D *h2 = new TH2D("TH1DExemplo", "Exemplo", 10, 0, 10, 20, 0, 20)
(TH2D *) 0x55a64c304ab0
root [1] for (Int_t i = 0; i < 1000; ++i) {Double_t x = gRandom->Uniform(0, 10);  Double_t y = gRandom->Uniform(0, 20);h2->Fill(x, y);}
root [2] h2->Draw()
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
root [3]
```



Protected Member Functions

TH2 ()
Constructor. More...

TH2 (const char *name, const char *title, Int_t nbinsx, Double_t xlow, Double_t xup, Int_t nbinsy, Double_t ylow, Double_t yup)
See comments in the TH1 base class constructors. More...

TH2 (const char *name, const char *title, Int_t nbinsx, const Double_t *xbins, Int_t nbinsy, Double_t ylow, Double_t yup)
See comments in the TH1 base class constructors. More...

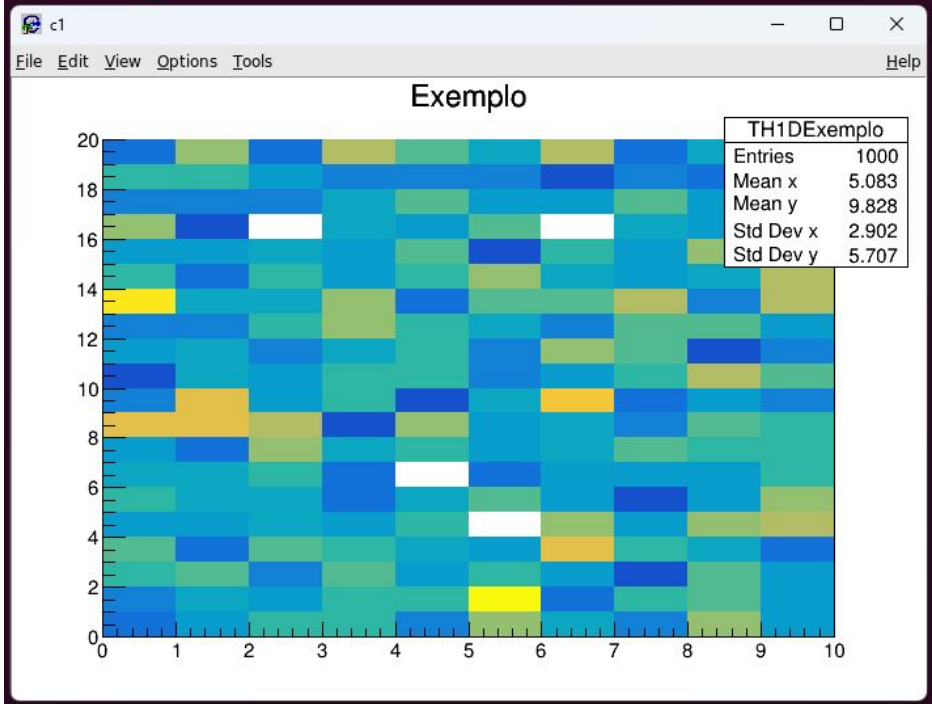
TH2 (const char *name, const char *title, Int_t nbinsx, Double_t xlow, Double_t xup, Int_t nbinsy, const Double_t *ybins)
See comments in the TH1 base class constructors. More...

TH2 (const char *name, const char *title, Int_t nbinsx, const Double_t *xbins, Int_t nbinsy, const Double_t *ybins)
See comments in the TH1 base class constructors. More...

TH2 (const char *name, const char *title, Int_t nbinsx, const Float_t *xbins, Int_t nbinsy, const Float_t *ybins)
See comments in the TH1 base class constructors. More...

Classes do ROOT: Histogramas - TH2

```
root [5] h2->Draw("COL")
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
root [6]
```



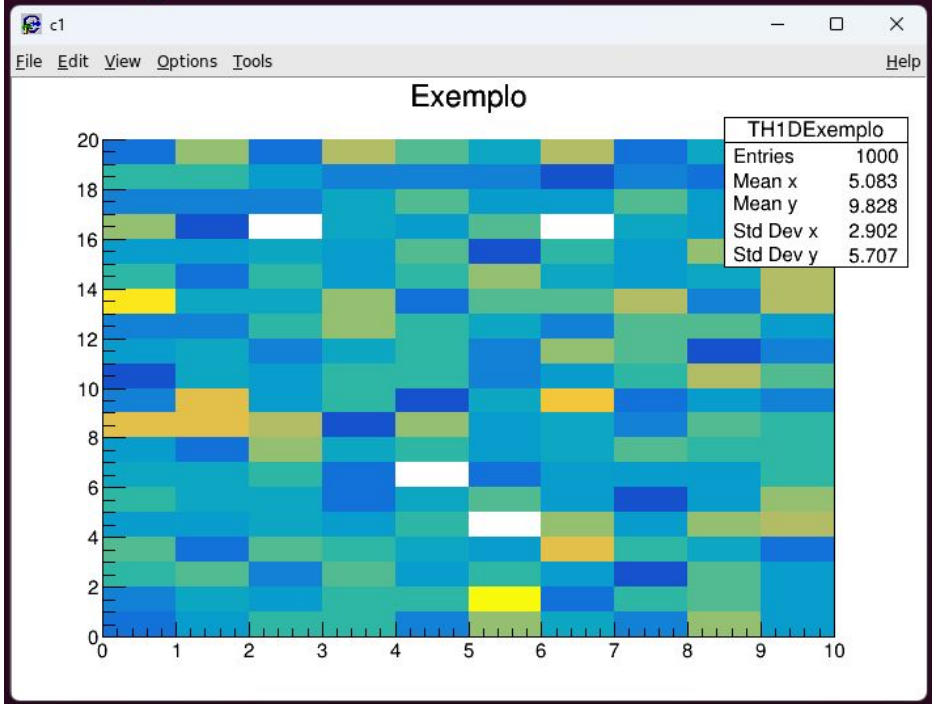
https://root.cern/doc/master/classTHistPainter.html

Options supported for 2D histograms

Option	Description
" "	Default (color plot).
"ARR"	Arrow mode. Shows gradient between adjacent cells.
"BOX"	A box is drawn for each cell with surface proportional to the content's absolute value. A negative content is marked with a X.
"BOX1"	A button is drawn for each cell with surface proportional to content's absolute value. A sunken button is drawn for negative values a raised one for positive.
"COL"	A box is drawn for each cell with a color scale varying with contents. All the none empty bins are painted. Empty bins are not painted unless some bins have a negative content because in that case the null bins might be not empty. TProfile2D histograms are handled differently because, for this type of 2D histograms, it is possible to know if an empty bin has been filled or not. So even if all the bins' contents are positive some empty bins might be painted. And vice versa, if some bins have a negative content some empty bins might be not painted (default).
"COLZ"	Same as "COL". In addition the color palette is also drawn.
"COL2"	Alternative rendering algorithm to "COL". Can significantly improve rendering performance for large, non-sparse 2-D histograms.
"COLZ2"	Same as "COL2". In addition the color palette is also drawn.
"Z CJUST"	In combination with colored options "COL","CONT0" etc: Justify labels in the color palette at color boundaries. For more details see TPaletteAxis
"CANDLE"	Draw a candle plot along X axis.
"CANDLEX"	Same as "CANDLE".
"CANDLEY"	Draw a candle plot along Y axis.
"CANDLEXn"	Draw a candle plot along X axis. Different candle-styles with n from 1 to 6.
"CANDLEYn"	Draw a candle plot along Y axis. Different candle-styles with n from 1 to 6.
"VIOLIN"	Draw a violin plot along X axis.
"VIOLINX"	Same as "VIOLIN".
"VIOLINY"	Draw a violin plot along Y axis.
"VIOLINXn"	Draw a violin plot along X axis. Different violin-styles with n being 1 or 2.
"VIOLINYn"	Draw a violin plot along Y axis. Different violin-styles with n being 1 or 2.
"CONT"	Draw a contour plot (same as CONT0).
"CONT0"	Draw a contour plot using surface colors to distinguish contours.
"CONT1"	Draw a contour plot using line styles to distinguish contours.
"CONT2"	Draw a contour plot using the same line style for all contours.
"CONT3"	Draw a contour plot using fill area colors.
"CONT4"	Draw a contour plot using surface colors (SURF option at theta = 0).
"LIST"	Generate a list of TGraph objects for each contour.
"SAME0"	Same as "SAME" but do not use the z-axis range of the first plot.
"SAMES0"	Same as "SAMES" but do not use the z-axis range of the first plot.

Classes do ROOT: Histogramas - TH2

```
root [5] h2->Draw("COL")
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
root [6]
```



https://root.cern/doc/master/classTHistPainter.html

Options supported for 2D histograms

Option	Description
" "	Default (color plot).
"ARR"	Arrow mode. Shows gradient between adjacent cells.
"BOX"	A box is drawn for each cell with surface proportional to the content's absolute value. A negative content is marked with a X.
"BOX1"	A button is drawn for each cell with surface proportional to content's absolute value. A sunken button is drawn for negative values a raised one for positive.
"COL"	A box is drawn for each cell with a color scale varying with contents. All the none empty bins are painted. Empty bins are not painted unless some bins have a negative content because in that case the null bins might be not empty. TProfile2D histograms are handled differently because, for this type of 2D histograms, it is possible to know if an empty bin has been filled or not. So even if all the bins' contents are positive some empty bins might be painted. And vice versa, if some bins have a negative content some empty bins might be not painted (default).
"COLZ"	Same as "COL". In addition the color palette is also drawn.
"COL2"	Alternative rendering algorithm to "COL". Can significantly improve rendering performance for large, non-sparse 2-D histograms.
"COLZ2"	Same as "COL2". In addition the color palette is also drawn.
"Z CJUST"	In combination with colored options "COL","CONT0" etc: Justify labels in the color palette at color boundaries. For more details see TPaletteAxis
"CANDLE"	Draw a candle plot along X axis.
"CANDLEX"	Same as "CANDLE".
"CANDLEY"	Draw a candle plot along Y axis.
"CANDLEXn"	Draw a candle plot along X axis. Different candle-styles with n from 1 to 6.
"CANDLEYn"	Draw a candle plot along Y axis. Different candle-styles with n from 1 to 6.
"VIOLIN"	Draw a violin plot along X axis.
"VIOLINX"	Same as "VIOLIN".
"VIOLINY"	Draw a violin plot along Y axis.
"VIOLINXn"	Draw a violin plot along X axis. Different violin-styles with n being 1 or 2.
"VIOLINYn"	Draw a violin plot along Y axis. Different violin-styles with n being 1 or 2.
"CONT"	Draw a contour plot (same as CONT0).
"CONT0"	Draw a contour plot using surface colors to distinguish contours.
"CONT1"	Draw a contour plot using line styles to distinguish contours.
"CONT2"	Draw a contour plot using the same line style for all contours.
"CONT3"	Draw a contour plot using fill area colors.
"CONT4"	Draw a contour plot using surface colors (SURF option at theta = 0).
"LIST"	Generate a list of TGraph objects for each contour.
"SAME0"	Same as "SAME" but do not use the z-axis range of the first plot.
"SAMES0"	Same as "SAMES" but do not use the z-axis range of the first plot.

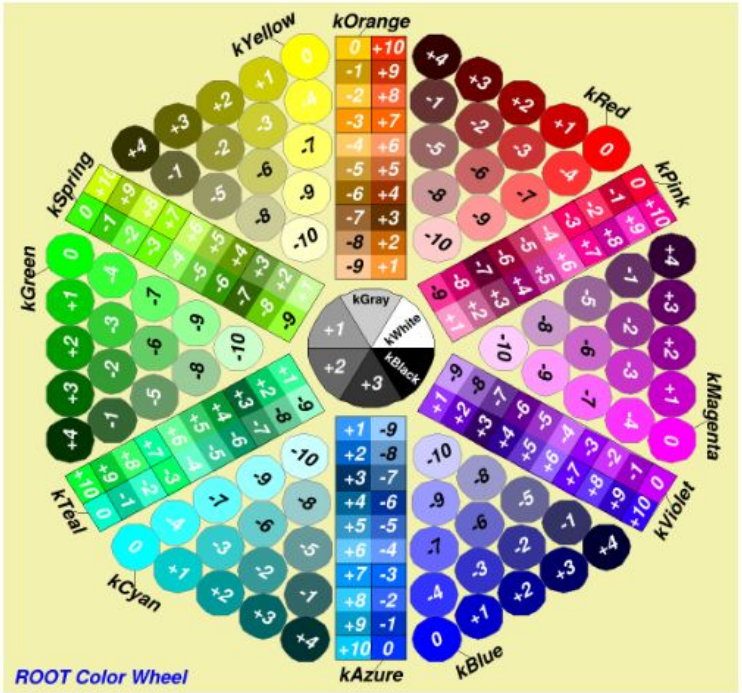
Classes do ROOT: THistPainter, TColor

https://root.cern/doc/master/classTHistPainter.html

https://root.cern.ch/doc/master/classTColor.html

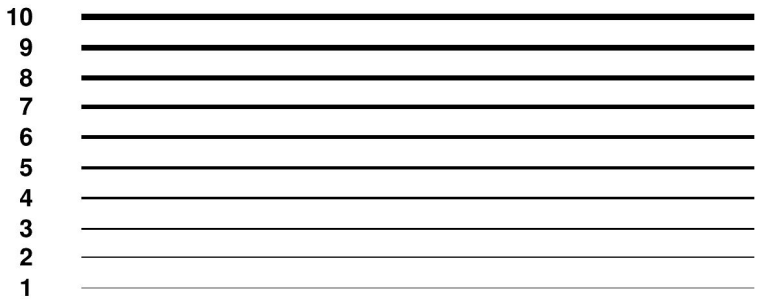
Options supported for 2D histograms

Option	Description
" "	Default (color plot).
"ARR"	Arrow mode. Shows gradient between adjacent cells.
"BOX"	A box is drawn for each cell with surface proportional to the content's absolute value. A negative content is marked with a X.
"BOX1"	A button is drawn for each cell with surface proportional to content's absolute value. A sunken button is drawn for negative values a raised one for positive.
"COL"	A box is drawn for each cell with a color scale varying with contents. All the none empty bins are painted. Empty bins are not painted unless some bins have a negative content because in that case the null bins might be not empty. `TProfile2D` histograms are handled differently because, for this type of 2D histograms, it is possible to know if an empty bin has been filled or not. So even if all the bins' contents are positive some empty bins might be painted. And vice versa, if some bins have a negative content some empty bins might be not painted (default).
"COLZ"	Same as "COL". In addition the color palette is also drawn.
"COL2"	Alternative rendering algorithm to "COL". Can significantly improve rendering performance for large, non-sparse 2-D histograms.
"COLZ2"	Same as "COL2". In addition the color palette is also drawn.
"Z CJUST"	In combination with colored options "COL","CONT0" etc: Justify labels in the color palette at color boundaries. For more details see `TPaletteAxis`
"CANDLE"	Draw a candle plot along X axis.
"CANDLEX"	Same as "CANDLE".
"CANDLEY"	Draw a candle plot along Y axis.
"CANDLEXn"	Draw a candle plot along X axis. Different candle-styles with n from 1 to 6.
"CANDLEYn"	Draw a candle plot along Y axis. Different candle-styles with n from 1 to 6.
"VIOLIN"	Draw a violin plot along X axis.
"VIOLINX"	Same as "VIOLIN".
"VIOLINY"	Draw a violin plot along Y axis.
"VIOLINXn"	Draw a violin plot along X axis. Different violin-styles with n being 1 or 2.
"VIOLINYn"	Draw a violin plot along Y axis. Different violin-styles with n being 1 or 2.
"CONT"	Draw a contour plot (same as CONT0).
"CONT0"	Draw a contour plot using surface colors to distinguish contours.
"CONT1"	Draw a contour plot using line styles to distinguish contours.
"CONT2"	Draw a contour plot using the same line style for all contours.
"CONT3"	Draw a contour plot using fill area colors.
"CONT4"	Draw a contour plot using surface colors (SURF option at theta = 0).
"LIST"	Generate a list of `TGraph` objects for each contour.
"SAME0"	Same as "SAME" but do not use the z-axis range of the first plot.
"SAMES0"	Same as "SAMES" but do not use the z-axis range of the first plot.

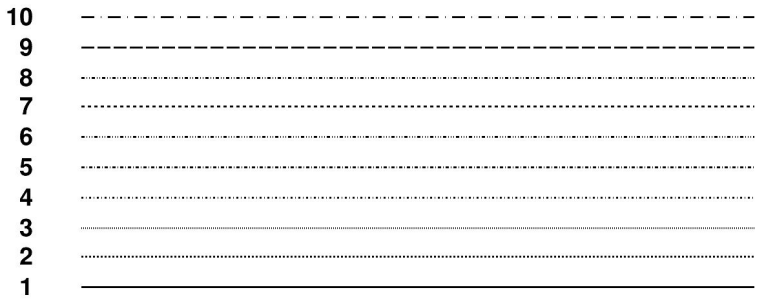


Classes do ROOT:

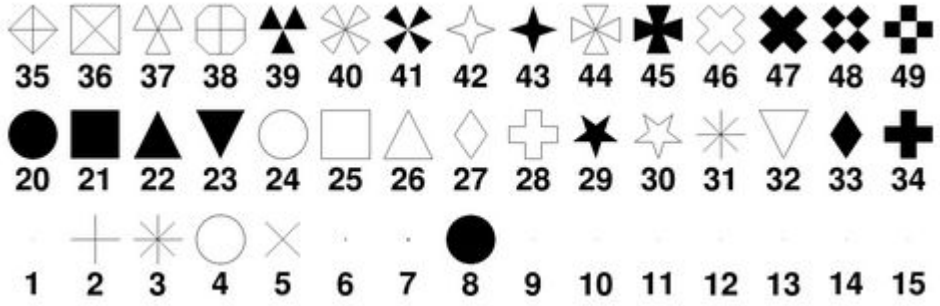
h1.SetLineWidth();



h1.SetLineStyle();



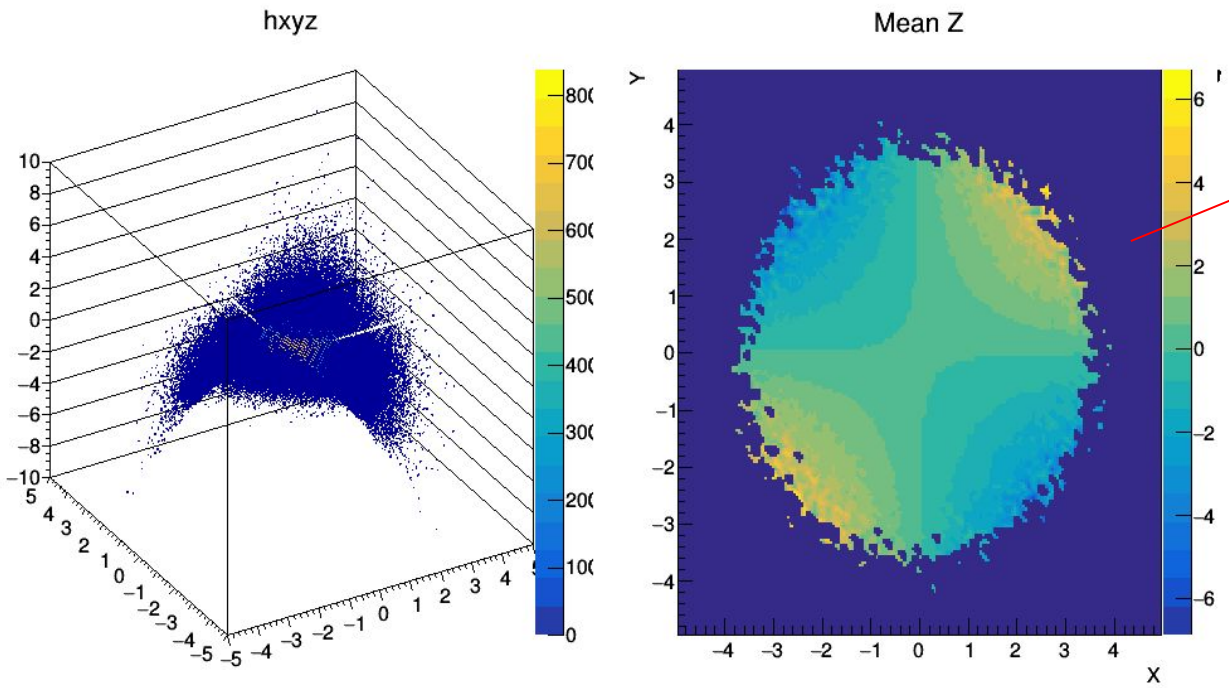
h1.SetMarkerStyle();



h1.SetFillColor();

Classes do ROOT: Histogramas - TH3

https://root.cern/doc/v612/classTH3.html



Função Project3DProfile que retorna um objeto da classe TProfile2D que por sua vez é um herdeiro da classe TH2D

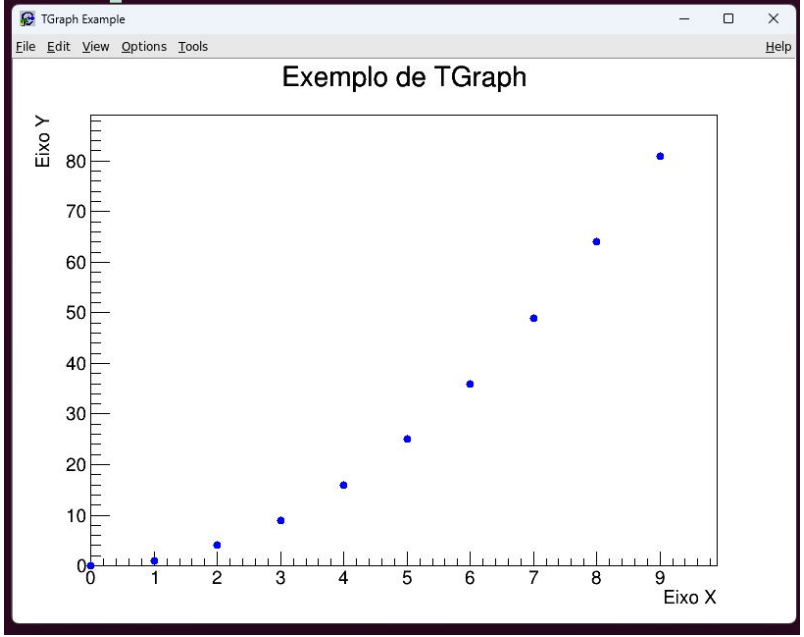
Classes do ROOT: Gráficos - TGraph

TGraph	A Graph is a graphics object made of two arrays X and Y with npoints each
TGraph2D	Graphics object made of three arrays X, Y and Z with the same number of points each
TGraph2DErrors	Graph 2D class with errors
TGraph2DPainter	The TGraphDelaunay painting class
TGraphAsymmErrors	TGraph with asymmetric error bars
TGraphBentErrors	A TGraphBentErrors is a TGraph with bent, assymetric error bars
TGraphDelaunay	TGraphDelaunay generates a Delaunay triangulation of a TGraph2D
TGraphDelaunay2D	TGraphDelaunay2D generates a Delaunay triangulation of a TGraph2D
TGraphEdge	An edge object connecting two nodes which can be added in a TGraphStruct
TGraphEditor	
TGraphErrors	A TGraphErrors is a TGraph with error bars
TGraphNode	A graph node object which can be added in a TGraphStruct
TGraphPainter	The graph painter class
TGraphPolar	To draw a polar graph
TGraphPolargram	To draw polar axis

Classes do ROOT: Gráficos - TGraph

https://root.cern/doc/v612/classTGraph.html

```
(base) mthiel@mauricio-uerj:~$ root -l
root [0] .x example_TGraph.C
root [1]
```



example_TGraph.C

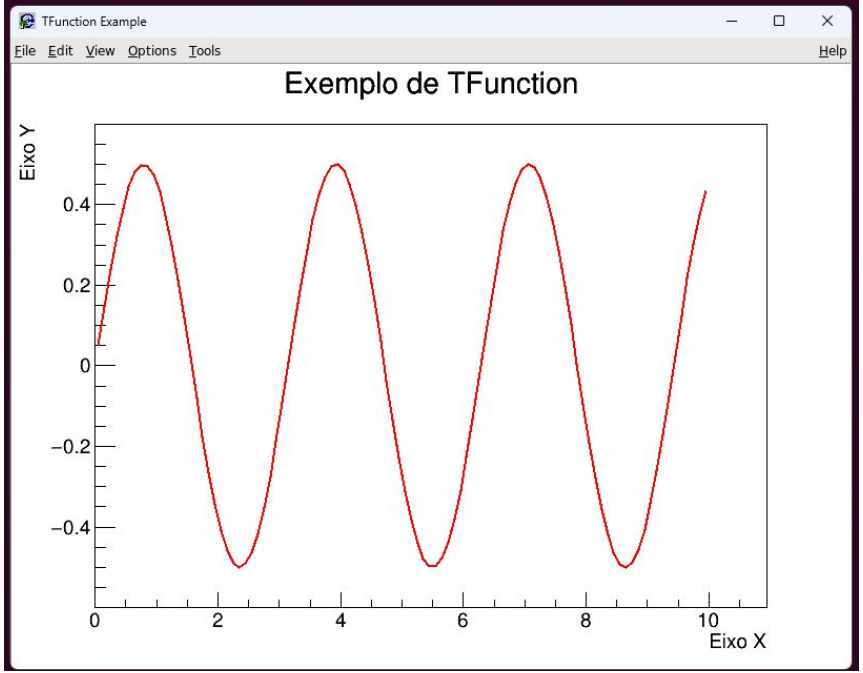
```
1 #include "TGraph.h"
2 #include "TCanvas.h"
3
4 void example_TGraph() {
5     // Dados para o gráfico
6     const Int_t nPoints = 10;
7     Double_t x[nPoints] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
8     Double_t y[nPoints] = {0, 1, 4, 9, 16, 25, 36, 49, 64, 81};
9
10    // Criando o TGraph
11    TGraph *graph = new TGraph(nPoints, x, y);
12
13    // Customizando o gráfico
14    graph->SetMarkerStyle(20); // Definindo o estilo dos marcadores
15    graph->SetMarkerColor(kBlue); // Definindo a cor dos marcadores
16    graph->SetTitle("Exemplo de TGraph"); // Título do gráfico
17    graph->GetXaxis()->SetTitle("Eixo X"); // Nome do eixo X
18    graph->GetYaxis()->SetTitle("Eixo Y"); // Nome do eixo Y
19
20    // Criando um canvas para o gráfico
21    TCanvas *c = new TCanvas("canvas", "TGraph Example", 800, 600);
22
23    // Desenhando o gráfico no canvas
24    graph->Draw("AP"); // "AP" para desenhar marcadores e conectar com linhas
25
26    // Mostrando o canvas
27    c->Draw();
28 }
```

Saving file "/home/mthiel/example_TGraph.C"... C++ Tab Width: 8 Ln 28, Col 2 INS

Classes do ROOT: Funções - TF1

https://root.cern/doc/v612/classTF1.html

```
(base) mthiel@mauricio-uerj:~$ root -l
root [0] .x example_TFunction.C
root [1]
```



example_TFunction.C

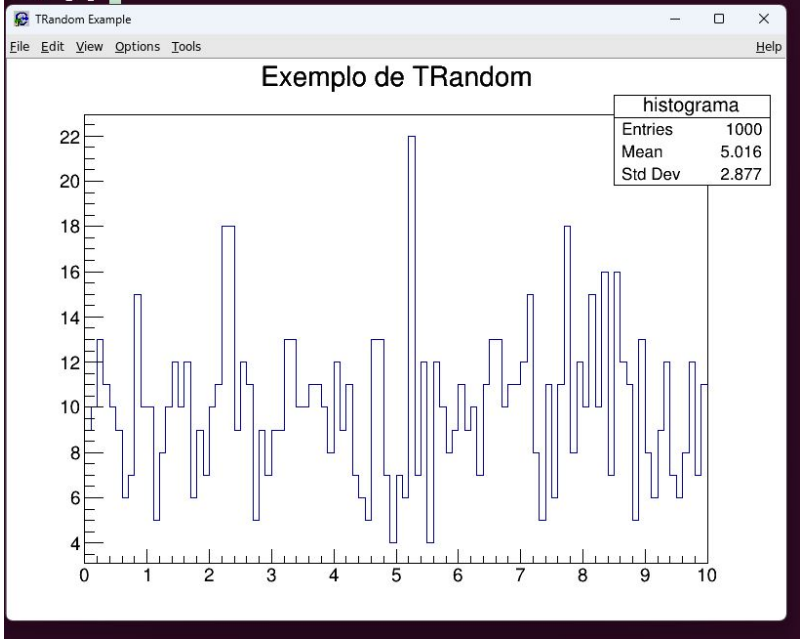
```cpp
#include "TCanvas.h"
#include "TGraph.h"
#include "TF1.h"

// Definição da função matemática
Double_t minhaFuncao(Double_t *x, Double_t *par) {
    return TMath::Sin(x[0]) * TMath::Cos(x[0]);
}

void example_TFunction() {
    // Criando um objeto TF1 com a função definida
    TF1 *f = new TF1("minhaFuncao", minhaFuncao, 0.0, 10.0, 0);

    // Criando um gráfico TGraph a partir do TF1
    TGraph *graph = new TGraph(f);

    // Customizando o gráfico
    graph->SetTitle("Exemplo de TFunction"); // Título do gráfico
    graph->GetXaxis()->SetTitle("Eixo X"); // Nome do eixo X
    graph->GetYaxis()->SetTitle("Eixo Y"); // Nome do eixo Y

    // Criando um canvas para o gráfico
    TCanvas *c = new TCanvas("canvas", "TFunction Example", 800, 600);

    // Desenhando o gráfico no canvas
    graph->Draw("AL"); // "AL" para desenhar a linha da função

    // Mostrando o canvas
    c->Draw();
}
```

Loading file "/home/mthiel/example_TFunction.C"... C++ Tab Width: 8 Ln 1, Col 1 INS

Classes do ROOT: Gerador de números aleatórios - TRandom

https://root.cern/doc/v612/classTRandom.html

```
(base) mthiel@mauricio-uerj: $ root -l
root [0]
root [0] .x example_TRandom.C
root [1]
```



example_TRandom.C
~/

Open Save

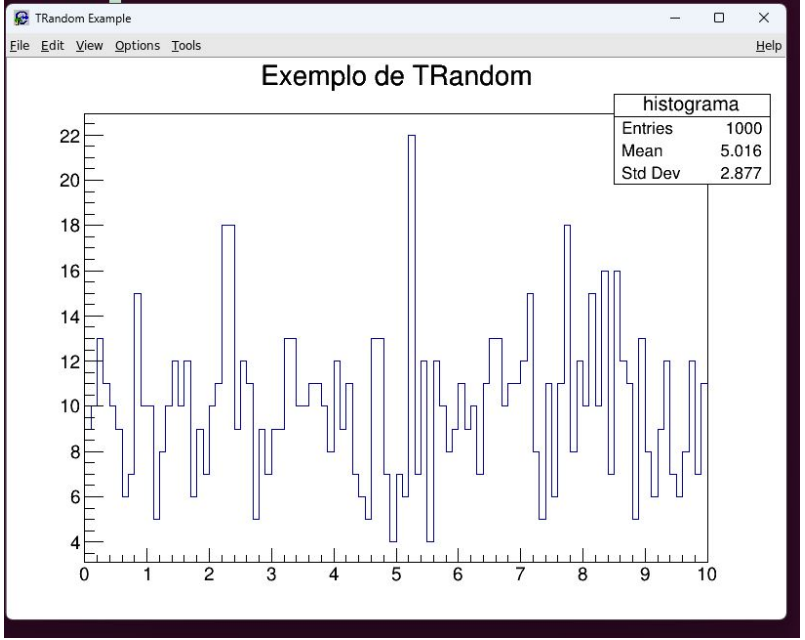
```
1 #include "TCanvas.h"
2 #include "TH1F.h"
3 #include "TRandom.h"
4
5 void example_TRandom() {
6     // Criando um objeto TRandom
7     TRandom *rand = new TRandom();
8
9     // Criando um histograma para armazenar os números aleatórios
10    TH1F *hist = new TH1F("histograma", "Exemplo de TRandom", 100, 0, 10);
11
12    // Preenchendo o histograma com números aleatórios
13    for (int i = 0; i < 1000; ++i) {
14        Double_t randomNumber = rand->Uniform(0, 10); // Gerando um número aleatório
  uniformemente distribuído entre 0 e 10
15        hist->Fill(randomNumber);
16    }
17
18    // Criando um canvas para o histograma
19    TCanvas *c = new TCanvas("canvas", "TRandom Example", 800, 600);
20
21    // Desenhando o histograma no canvas
22    hist->Draw();
23
24    // Mostrando o canvas
25    c->Draw();
26 }
```

C++ Tab Width: 8 Ln 27, Col 1 INS

Classes do ROOT: Gerador de números aleatórios - TRandom

https://root.cern/doc/v612/classTRandom.html

```
(base) mthiel@mauricio-uerj: $ root -l
root [0]
root [0] .x example_TRandom.C
root [1]
```



Open | example_TRandom.C ~/ | Save

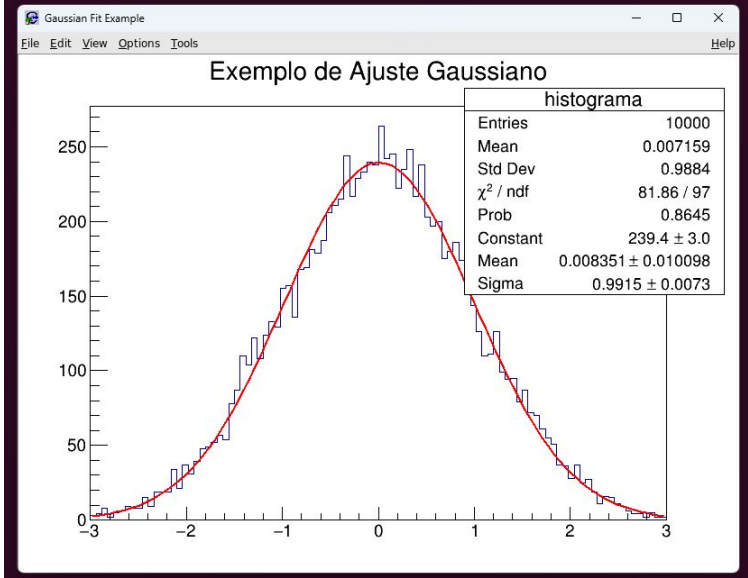
```
1  #include "TCanvas.h"
2  #include "TH1F.h"
3  #include "TRandom.h"
4
5  void example_TRandom() {
6      // Criando um objeto TRandom
7      TRandom *rand = new TRandom();
8
9      // Criando um histograma para armazenar os números aleatórios
10     TH1F *hist = new TH1F("histograma", "Exemplo de TRandom", 100, 0, 10);
11
12     // Preenchendo o histograma com números aleatórios
13     for (int i = 0; i < 1000; ++i) {
14         Double_t randomNumber = rand->Uniform(0, 10); // Gerando um número aleatório
   uniformemente distribuído entre 0 e 10
15         hist->Fill(randomNumber);
16     }
17
18     // Criando um canvas para o histograma
19     TCanvas *c = new TCanvas("canvas", "TRandom Example", 800, 600);
20
21     // Desenhando o histograma no canvas
22     hist->Draw();
23
24     // Mostrando o canvas
25     c->Draw();
26 }
```

C++ | Tab Width: 8 | Ln 27, Col 1 | INS

Será muito utilizado na aula sobre simulações a Monte Carlo

~~Classes~~ do ROOT: Fit função da classe TH1

```
(base) mthiel@mauricio-uerj:~$ root -l
root [0] .x example_GaussianFit.C
 FCN=81.8616 FROM MIGRAD    STATUS=CONVERGED      61 CALLS          62 TOTAL
                     EDM=5.36285e-08    STRATEGY= 1      ERROR MATRIX ACCURATE
  EXT PARAMETER                                   STEP         FIRST
  NO.   NAME      VALUE            ERROR          SIZE      DERIVATIVE
   1  Constant     2.39446e+02   2.95448e+00   1.06995e-02   1.17823e-04
   2  Mean         8.35132e-03   1.00982e-02   4.48651e-05   1.61940e-02
   3  Sigma        9.91538e-01   7.27314e-03   8.86835e-06   8.06804e-02
root [1]
```



Open *example_GaussianFit.C ~/ Save

```
1 #include "TCanvas.h"
2 #include "TH1F.h"
3 #include "TF1.h"
4 #include "TRandom.h"
5 #include "TStyle.h"
6
7 void example_GaussianFit() {
8     // Criando um objeto TRandom
9     TRandom *rand = new TRandom();
10
11    // Criando um histograma para armazenar os números aleatórios
12    TH1F *hist = new TH1F("histograma", "Exemplo de Ajuste Gaussiano", 100, -3, 3);
13
14    // Gerando e preenchendo o histograma com números aleatórios de uma distribuição gaussiana
15    for (int i = 0; i < 10000; ++i) {
16        Double_t randomNumber = rand->Gaus(0, 1); // Gerando um número aleatório de uma
  distribuição gaussiana com média 0 e desvio padrão 1
17        hist->Fill(randomNumber);
18    }
19
20    // Criando um canvas para o histograma
21    TCanvas *c = new TCanvas("canvas", "Gaussian Fit Example", 800, 600);
22
23    // Desenhando o histograma no canvas
24    hist->Draw();
25
26    // Criando uma função gaussiana para o fit
27    TF1 *gaussianFit = new TF1("gaussianFit", "gaus", -3, 3);
28
29    // Fazendo o fit do histograma com a função gaussiana
30    hist->Fit(gaussianFit);
31
32    // Customizando as estatísticas do fit
33    gStyle->SetOptFit(1111); // Mostra todas as informações do fit
34
35    // Mostrando o canvas com o histograma e o fit
36    c->Draw();
37 }
38
```

C++ Tab Width: 8 Ln 38, Col 1 INS

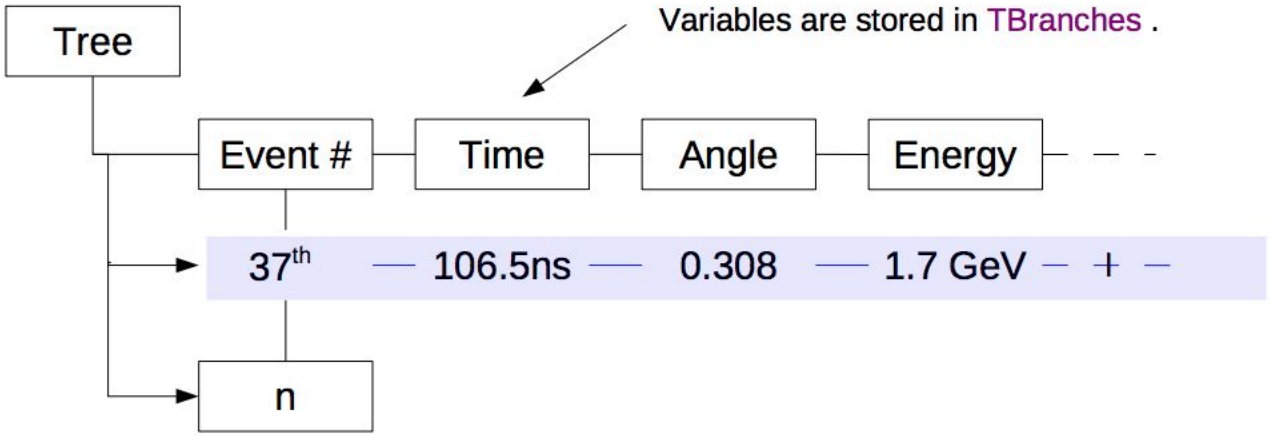
Classes do ROOT: Outros objetos

- **TGraphAsymmErrors** - Classe para gráficos 1D com erros assimétricos nos pontos
- **TFile** - Classe para criar, ler e escrever arquivos ROOT
- **TTree** - Classe para armazenar grandes quantidades de dados em forma de uma árvore
- **TLegend** - Classe para criar legendas em gráficos e histogramas
- **TPad** - Classe para criar e controlar regiões em um TCanvas onde gráficos e histogramas podem ser desenhados
- **THStack** - Classe para empilhar histogramas verticalmente em um TCanvas
- **TMath** - Biblioteca matemática com funções matemáticas comuns, como seno, cosseno, exponencial, etc
- **TStyle** - Classe para controlar o estilo de plotagem, incluindo cores, fontes, tamanhos, etc
- **TGaxis** - Classe para controlar e customizar os eixos de um gráfico ou histograma
- **TMinuit** - Classe para realizar minimização de funções, útil para ajuste de curvas

TTree

https://root.cern.ch/doc/master/classTTree.html

- Uma TTree é uma estrutura de dados para organizar e manipular várias variáveis de dados ao mesmo tempo
- Capaz de criar histogramas instantaneamente, incluindo fazer cortes de seleção nos dados
- Usa os algoritmos internos de compressão do ROOT para reduzir o tamanho dos dados
    - Muito útil para armazenamento de dados



ROOT Command Line: TTree Example

```
Troot [0] TFile *f1 = new TFile("tree.root");
root [1] f1->ls();
TFile**         tree.root
 TFile*         tree.root
  KEY: TTree    tree1;1 Reconstruction ntuple
root [2] TTree *mytree = (TTree *)f1->Get("tree1");
root [3] mytree->Print();
******************************************************************************
*Tree    :tree1     : Reconstruction ntuple                                  *
*Entries :   100000 : Total =         2810073 bytes  File  Size =    2171135 *
*        :          : Tree compression factor =   1.30                       *
******************************************************************************
*Br    0 :event     : event/I                                                *
*Entries :   100000 : Total  Size=     421248 bytes  File Size  =     134514 *
*Baskets :       12 : Basket Size=      32000 bytes  Compression=   2.85     *
*.............................................................................*
*Br    1 :ebeam     : ebeam/F                                                *
*Entries :   100000 : Total  Size=     421248 bytes  File Size  =     260330 *
*Baskets :       12 : Basket Size=      32000 bytes  Compression=   1.47     *
*.............................................................................*
*Br    2 :px        : px/F                                                   *
*Entries :   100000 : Total  Size=     421194 bytes  File Size  =     359238 *
*Baskets :       12 : Basket Size=      32000 bytes  Compression=   1.07     *
*.............................................................................*
*Br    3 :py        : py/F                                                   *
*Entries :   100000 : Total  Size=     421194 bytes  File Size  =     359138 *
*Baskets :       12 : Basket Size=      32000 bytes  Compression=   1.07     *
*.............................................................................*
```

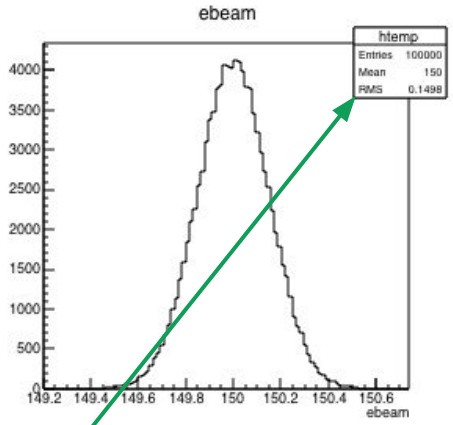
Create pointer to “tree1”

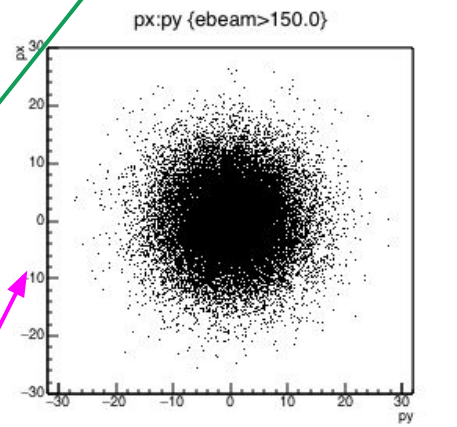
Print structure of tree to screen
This tree contains 7 variables:
`event, ebeam, px, py, pz, zv, chi2`

Turn on statistics box

```
root [4] TCanvas *c2 = new TCanvas("c2", "Tree canvas", 300, 600);
root [5] c2->Divide(1,2);
root [6] c2->cd(1);
root [7] gStyle->SetOptStat(1);
root [8] mytree->Draw("ebeam");
root [9] c2->cd(2);
root [10] mytree->Draw("px:py", "ebeam>150.0");
```

Draw scatter plot (`py` vs `px`) for events with `ebeam` > 150.

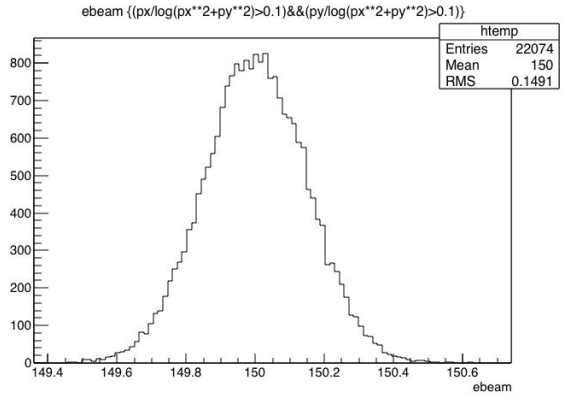




ROOT TTree: Complicated cuts

- Consider encapsulating your cuts as TCut objects
- TCut objects can be combined using C-style operators as usual
- They can be combined with other string cuts

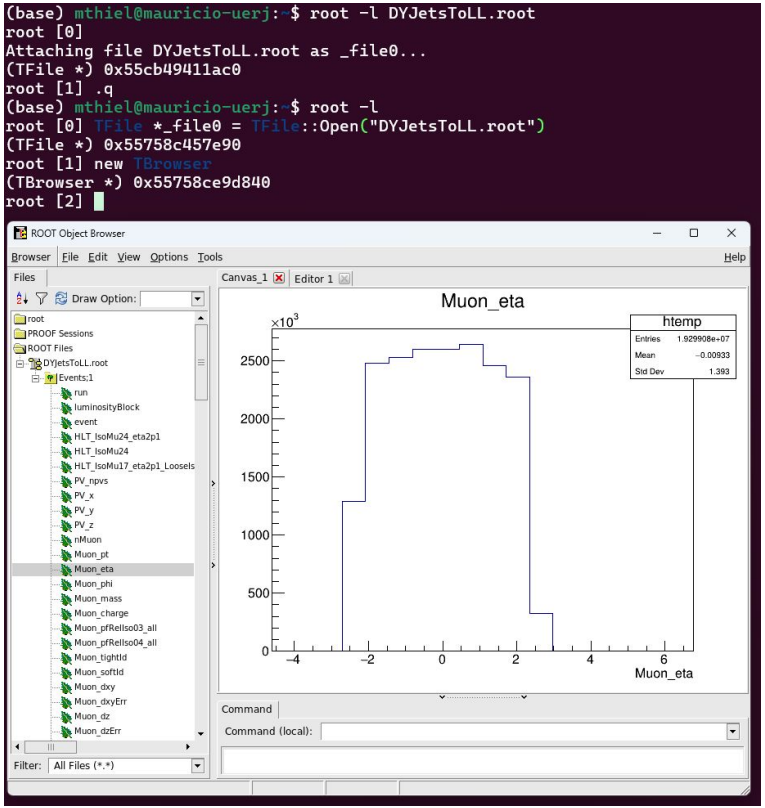
```
root [14] TCut * px_plane = new TCut(“px / log(px*2 + py**2) > 0.10”);
root [15] TCut * py_plane = new TCut(“py / log(px*2 + py**2) > 0.10”);
root [16] mytree->Draw(“ebeam”, *px_plane && *py_plane);
```



TTree

https://root.cern.ch/doc/master/classTTree.html

- https://opendata.cern.ch/record/12353
- Arquivo root com uma TTree
- Também conhecido como NTuple



TTree - MakeClass()

https://root.cern.ch/doc/master/classTTree.html

MakeClass()

```
Int_t TTree::MakeClass ( const char *  classname = nullptr,
                         Option_t *    option = ""
                       )                                          virtual
```

Generate a skeleton analysis class for this tree.

The following files are produced: classname.h and classname.C. If classname is 0, classname will be called "nameoftree".

The generated code in classname.h includes the following:

- Identification of the original tree and the input file name.
- Definition of an analysis class (data members and member functions).
- The following member functions:
  - constructor (by default opening the tree file),
  - GetEntry(Long64_t entry),
  - Init(TTree* tree) to initialize a new **TTree**,
  - Show(Long64_t entry) to read and dump entry.

The generated code in classname.C includes only the main analysis function Loop.

To use this function:

- Open your tree file (eg: **TFile** f("myfile.root");)
- T->MakeClass("MyClass");

where T is the name of the **TTree** in file myfile.root, and MyClass.h, MyClass.C the name of the files created by this function. In a **ROOT** session, you can do:

```
root > .L MyClass.C
root > MyClass* t = new MyClass;
root > t->GetEntry(12); // Fill data members of t with entry number 12.
root > t->Show();       // Show values of entry 12.
root > t->Show(16);     // Read and show values of entry 16.
root > t->Loop();       // Loop on all entries.
```

```
(base) mthiel@mauricio-uerj:~$ root -l DYJetsToLL.root
root [0]
Attaching file DYJetsToLL.root as _file0...
(TFile *) 0x55eee3228c70
root [1] _file0->ls()
TFile**         DYJetsToLL.root
 TFile*         DYJetsToLL.root
  KEY: TTree    Events;1        Events
```

```
root [2] TTree *tree = (TTree*)_file0->Get("Events")
(TTree *) 0x55eee3802c80
root [3] tree->MakeClass("MyTreeClass");
Info in <TTreePlayer::MakeClass>: Files: MyTreeClass.h and MyTreeClass.C generated from TTree: Events
root [4]
```

```
(base) mthiel@mauricio-uerj:~$ root -l
root [0] .L MyTreeClass.C
root [1] MyTreeClass l;
root [2] l.Loop()
```

TTree - MakeClass()

https://root.cern.ch/doc/master/classTTree.html

Exercício: Com a ntupla encontrada nas amostras de dados de seu grupo, use o MakeClass e faça o histograma de algumas distribuições.
Dica: https://youtu.be/P9aKV7W5zxg

Exercises

1. Create a function with parameters, p0 * sin (p1 * x) / x, and also draw it for different parameter values. Set the colour of the parametric function to blue. After having drawn the function, compute for the parameter values (p0 = 1, p1 = 2):
    a. Function value for x=1
    b. Function derivative for x=1
    c. Integral of the function between 0 and 3
2. Suppose you have this set of points defined in the attached file graphdata.txt. Plot these points using the TGraph class. Use as marker point a black box. Looking at the possible options for drawing the TGraph in TGraphPainter, plot a line connecting the points. Make a TGraphError and display it by using the attached data set, graphdata_error.txt, containing error in x and y.
3. Create a one-dimensional histogram with 50 bins between 0 to 10, and fill it with 10000 gaussian distributed random numbers with mean 5 and sigma 2. Plot the histogram and, looking at the documentation in the THistPainter, show in the statistic box the number of entries, the mean, the RMS, the integral of the histogram, the number of underflows, the number of overflows, the skewness and the kurtosis.
4. Using the tree contained in tree.root make a distribution of the total momentum of each whose beam energy was outside of the mean by more than 0.2. Use TCut objects to make your events selections. Project this distribution into a histogram, draw it and save it to a file.

backup

The Jupyter Notebook

A web-based interactive computing platform that combines code, equations, text and visualisations.

Many supported languages: C++, Python, Haskell, Julia…
One generally speaks about a “kernel” for a specific language

In a nutshell: an “interactive shell opened within the browser”

http://www.jupyter.org

ROOT interfaces on Jupyter notebook

- ROOT is well integrated with Jupyter Notebook, both for what concerns its Python and C++ interface
- What is Jupyter Notebook? https://jupyter.org/
  - Language of choice, share notebooks, interactive output, big data integration
- **How to integrate Jupyter notebook and ROOT:**
  - Install ROOT6 (> 6.05)
  - Install dependencies: pip install jupyter metakernel
  - Set up the ROOT environment (`. $ROOTSYS/bin/thisroot.[c]sh`) and then type in your shell: `root --notebook`

How It Looks Like

Text

Code

Graphics

Access TTree in Python using PyROOT and fill a histogram

Loop over the TTree called "events" in a file located on the web. The tree is accessed with the dot operator. Same holds for the access to the branches: no need to set them up - they are just accessed by name, again with the dot operator.

In [1]:
```
import ROOT

f = ROOT.TFile.Open("http://indico.cern.ch/event/395198/material/0/0.root");
h = ROOT.TH1F("TracksPt","Tracks;Pt [GeV/c];#",128,0,64)
for event in f.events:
    for track in event.tracks:
        h.Fill(track.Pt())
c = ROOT.TCanvas()
h.Draw()
c.Draw()
```

To execute the code, click shift+enter