

ROOT

warsztaty

Część 1

Strony internetowe

- **instrukcja instalacji dla różnych dystrybucji Linuxa**

root.cern/install/#download-a-pre-compiled-binary-distribution

- **pliki, których dziś będziemy używać:**

<https://github.com/KrzysztofProscinski/ROOT>

- **szerszy poradnik ROOT-a**

<https://www.fuw.edu.pl/~kpias/>

Dydaktyka -> Computer Tools for Nuclear Physics

Linux - powtórzenie

Komendy te należy wpisywać w linuxowy terminal.

- | | |
|------------------------------|--------------------------------------|
| > mkdir [folder] | - utworzenie nowego folderu |
| > touch [plik] | - utworzenie nowego pliku |
| > cd [folder] | - przejście do danego folderu |
| > cd .. | - przejście do folderu macierzystego |
| > cd | - przejście do folderu domowego |
| > ls | - zawartość obecnego folderu |
| > mv [plik] [folder] | - przeniesienie pliku do folderu |
| > mv [plik1] [plik2] | - zmiana nazwy pliku |
| > cp [plik] [folder]/ | - skopiowanie pliku do folderu |
| > cp -r [folder1] [folder2]/ | - skopiowanie folderu do folderu |
| > rm [plik] | - usunięcie pliku |

Komendy do terminala będą oznaczane jako "> [treść komendy]". Przykładowo "> cd" oznacza, że w terminalu należy wpisać "cd".

"-r" trzeba dodawać zawsze do operacji na folderach.

.bashrc

> cd

> nano ~/.bashrc

wpisać:

“export ROOTSYS=\$HOME/root

export PATH=\$PATH:\$ROOTSYS/bin

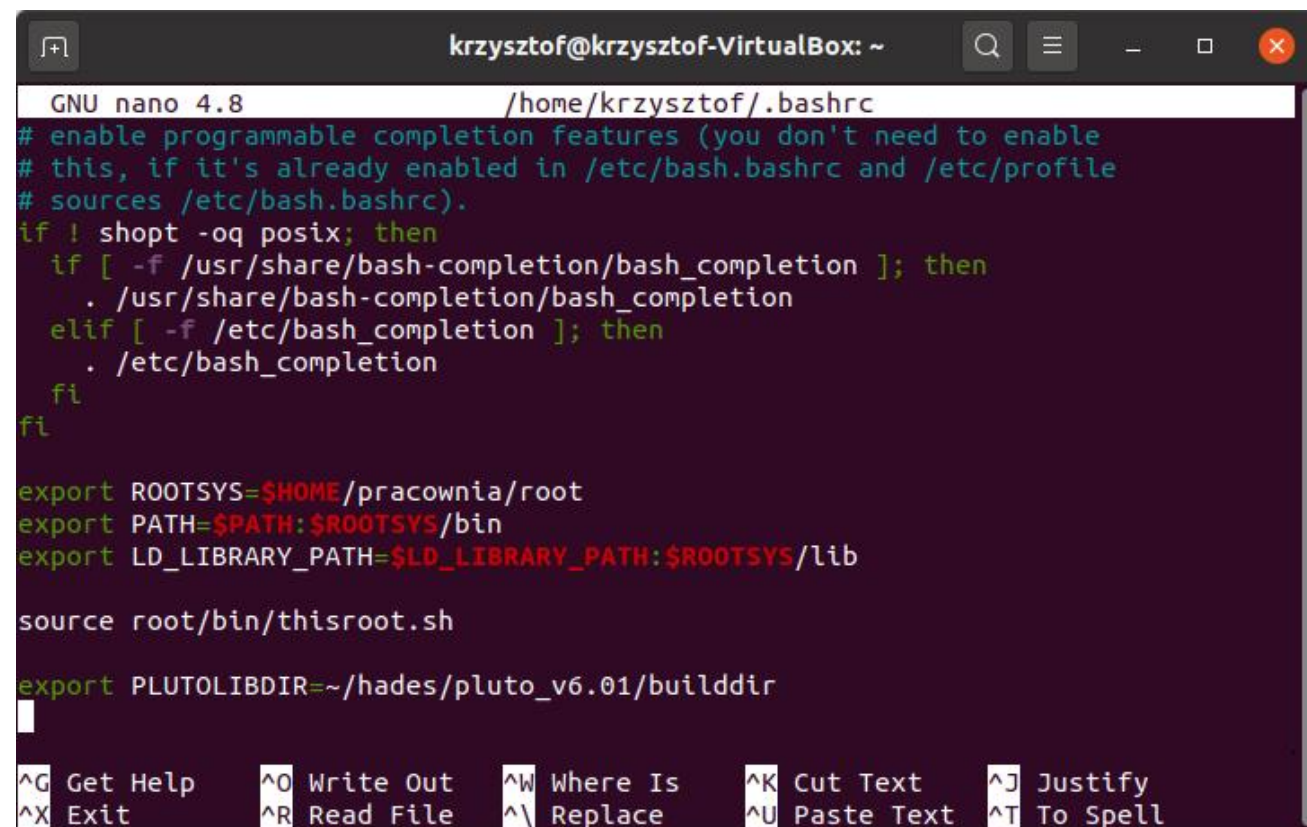
export

LD_LIBRARY_PATH=\$LD_LIBRARY_PATH:
\$ROOTSYS/lib

source root/bin/thisroot.sh”

.bashrc to plik domyślnie istniejący w Linuxach, do którego należy dopisać poniższe linijki, aby ROOT poprawnie działał.

“nano” to przykładowy edytor tekstowy w Linuxie. Zamiast tego można użyć innego edytora.



```
krzysztof@krzysztof-VirtualBox: ~
GNU nano 4.8 /home/krzysztof/.bashrc
# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
  if [ -f /usr/share/bash-completion/bash_completion ]; then
    . /usr/share/bash-completion/bash_completion
  elif [ -f /etc/bash_completion ]; then
    . /etc/bash_completion
  fi
fi

export ROOTSYS=$HOME/pracownia/root
export PATH=$PATH:$ROOTSYS/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ROOTSYS/lib

source root/bin/thisroot.sh

export PLUTOLIBDIR=~/.hades/pluto_v6.01/buildldir
^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify
^X Exit      ^R Read File  ^\ Replace   ^U Paste Text ^T To Spell
```

Niebieskie teksty w cudzysłowach oznaczają treści makr.

Jak uruchomić ROOT-a

- **uruchamianie**

> root

- **opcje uruchamiania**

> root -l -b

“-l” - bez ekranu powitalnego

“-b” - bez grafiki

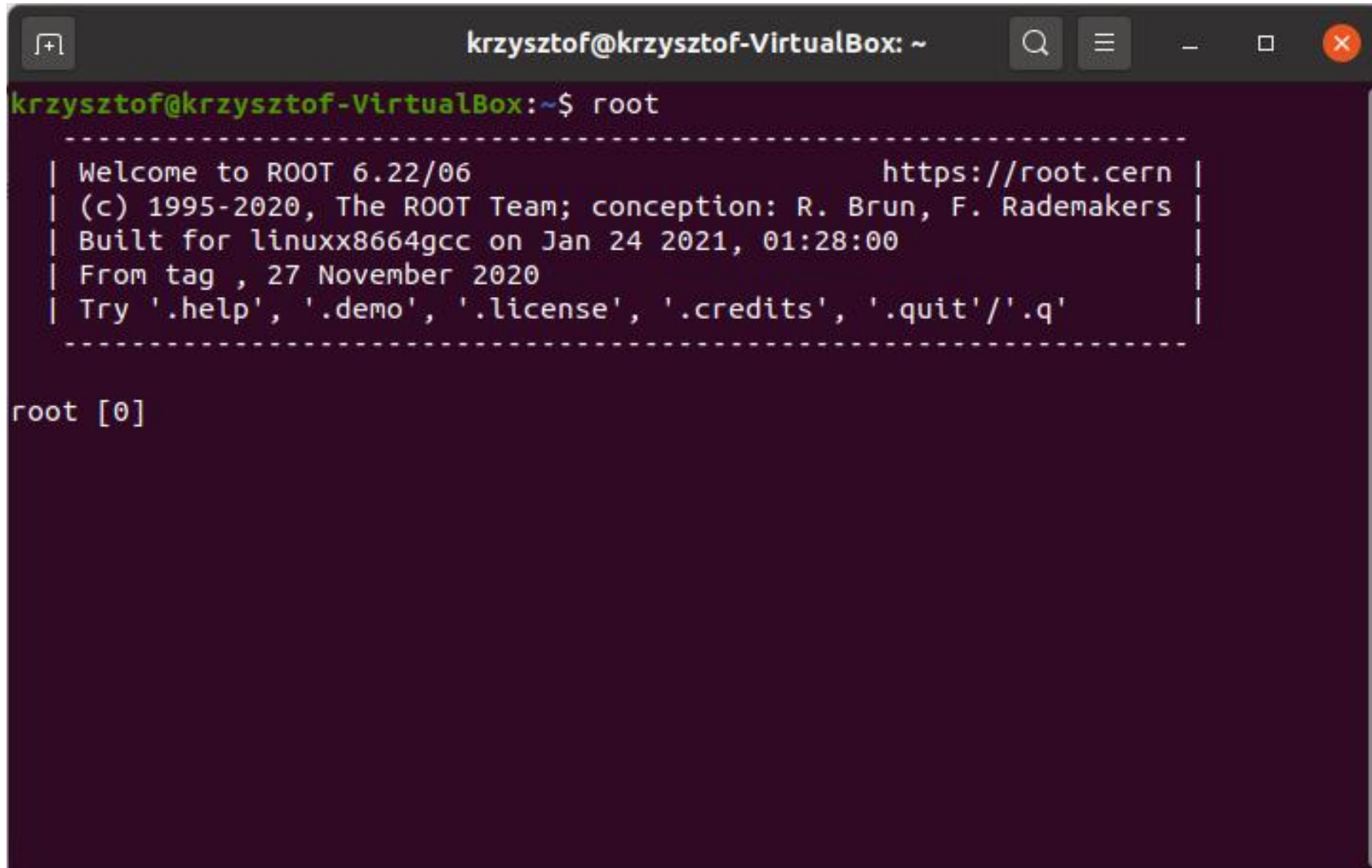
- **zamykanie**

> .q

Opcje można ze sobą składać, tzn. jeżeli zostaną wpisane obie, to obie zostaną uwzględnione.

Po uruchomieniu ROOT-a normalne linuxowe komendy (np. zmiana katalogu) są niedostępne. Aby móc ich użyć należy zamknąć ROOT-a.

Jak uruchomić ROOT-a

A terminal window titled 'krzysztof@krzysztof-VirtualBox: ~' with standard window controls. The prompt 'krzysztof@krzysztof-VirtualBox:~\$' is followed by the command 'root'. The output is a multi-line message enclosed in dashed lines, providing welcome information and usage instructions for ROOT 6.22/06. Below the dashed box, the prompt changes to 'root [0]'.

```
krzysztof@krzysztof-VirtualBox:~$ root
-----
| Welcome to ROOT 6.22/06                                     https://root.cern |
| (c) 1995-2020, The ROOT Team; conception: R. Brun, F. Rademakers |
| Built for linuxx8664gcc on Jan 24 2021, 01:28:00              |
| From tag , 27 November 2020                                  |
| Try '.help', '.demo', '.license', '.credits', '.quit'/'.q'   |
|-----|
root [0]
```

Zdalne połączenie (Windows)

- **wymagane programy**

PuTTY (lub inna aplikacja do zdalnego logowania)

Xming (lub inny serwer systemu X Windows)

- **uruchamianie**

uruchomić PuTTY

wpisać odpowiedni adres (lub adres IP) w pole "Host Name"

wcisnąć "Open"

- **dostęp do aplikacji graficznych**

uruchomić Xming, a następnie PuTTY

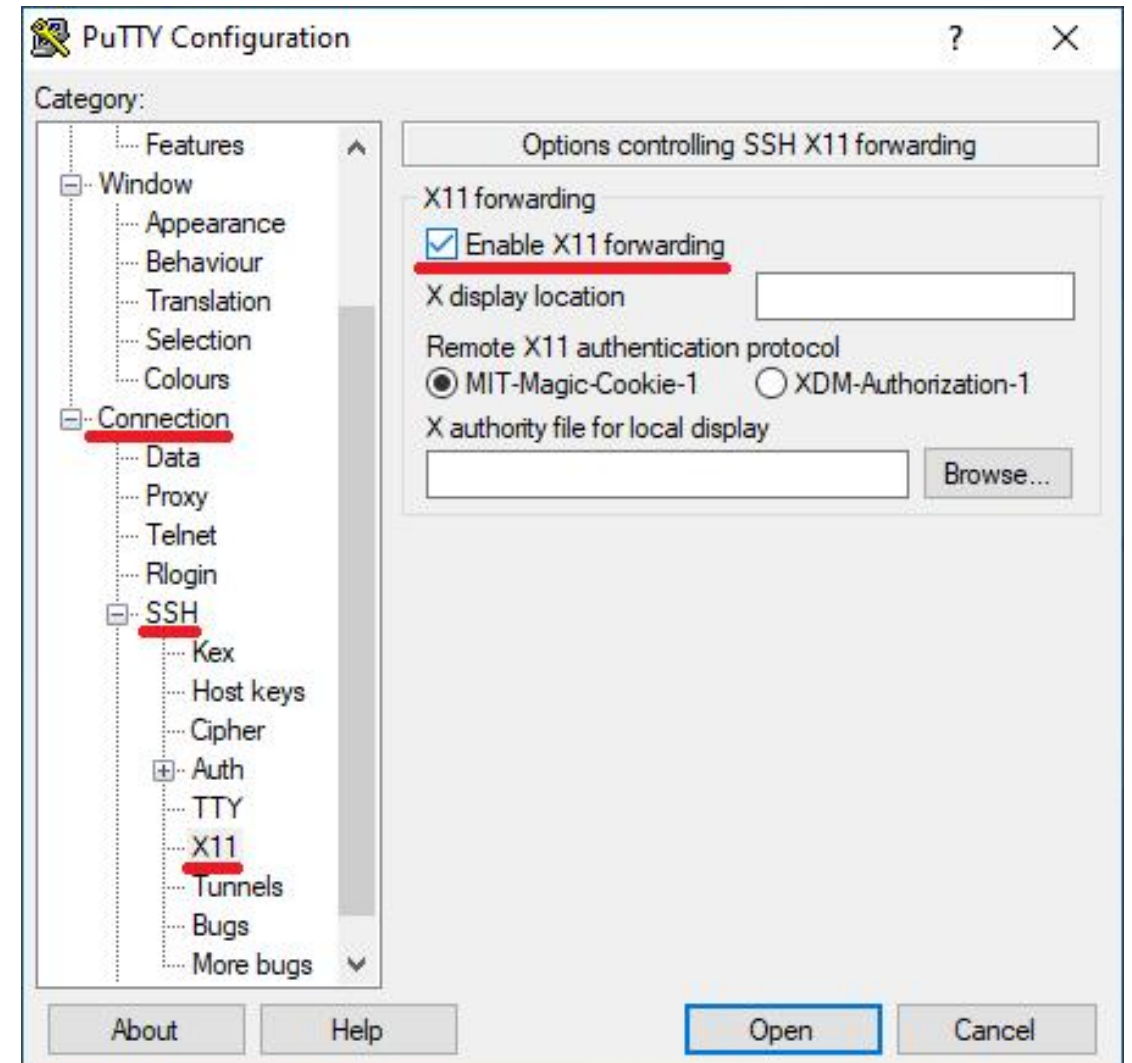
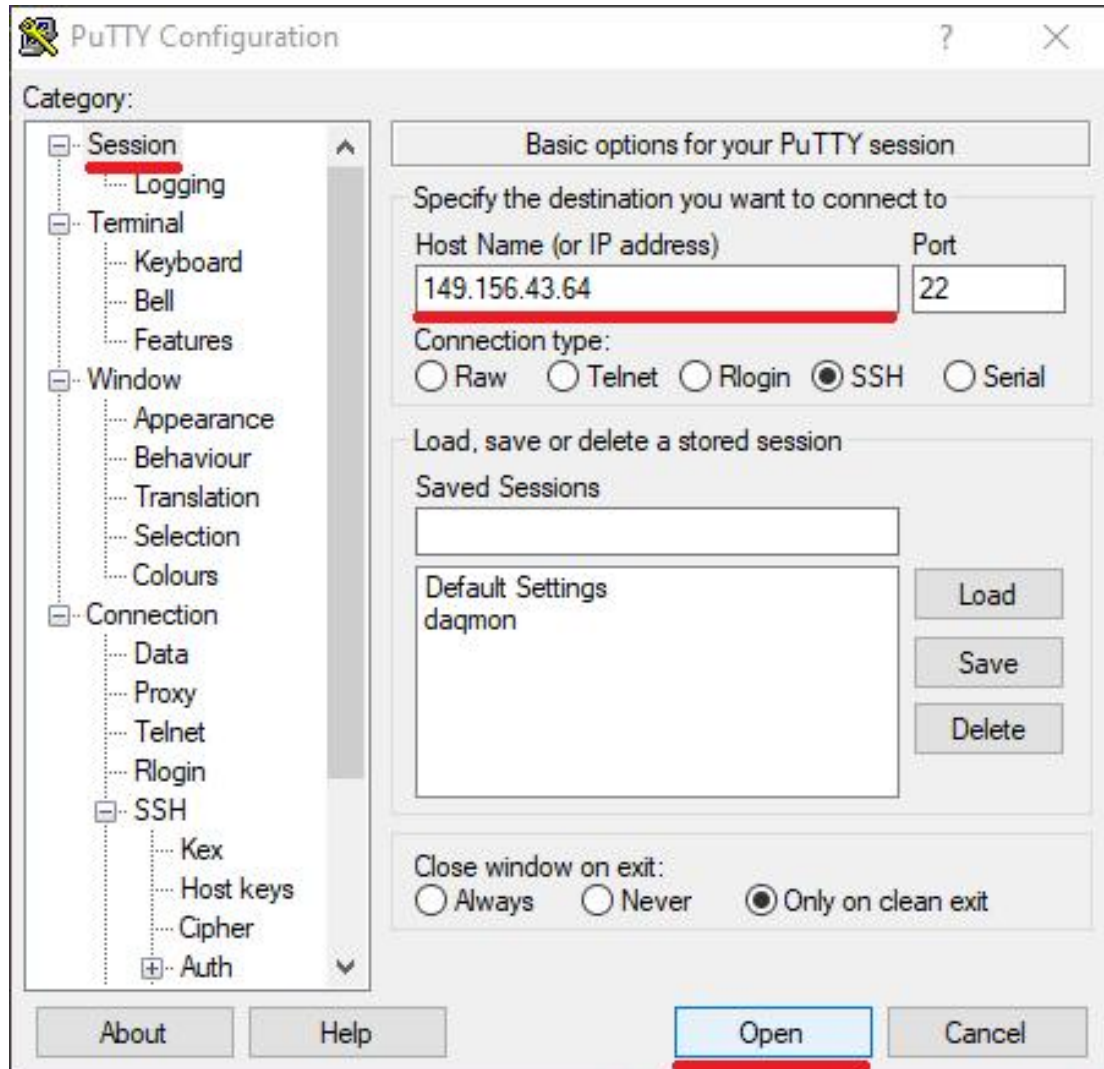
włączyć opcję "Enable X11 forwarding" (Connection->SSH->X11)

dalej jak przy normalnym uruchamianiu

Przykładowo adres
Studenckiej Pracowni
Komputerowej to
spk-ssh.if.uj.edu.pl,
a adres IP to 149.156.43.64.

Na niektórych zdalnych
komputerach użycie aplikacji
graficznych może nie być
możliwe i pozostaje jedynie
korzystanie z terminala.

Zdalne połączenie (Windows)



Zdalne połączenie (Linux)

- **połączenie**

> ssh [login]@[serwer]

przykład

> ssh kproscin@149.156.43.64

- **kopiowanie plików między komputerami**

> scp kproscin@149.156.43.64:~/plik.txt .

> scp plik.txt kproscin@149.156.43.64:~/.

- **zamykanie połączenia**

> exit

Polecenia scp należy wpisać na komputerze użytkownika, nie na komputerze zdalnym.

Do kopiowania plików należy wpisać "scp [adres pliku] [adres docelowy]".

<- W pierwszym przypadku kopiowany jest plik "plik.txt" z komputera pracowni na komputer użytkownika. Kropka wpisana zamiast adresu docelowego oznacza, że adresem docelowym jest aktualnie otwarty w terminalu folder.

W drugim przypadku kopiowany jest plik z komputera użytkownika na komputer pracowni. "~/" oznacza, że adresem docelowym jest folder domowy.

rootlogon.C

- makro ładowane przy każdym uruchomieniu ROOT-a

> nano rootlogon.C

wpisać

```
"#include<iostream>  
Bool_t rootlogon(void){  
cout<<"hello"<<endl;  
return kTRUE;  
}"
```

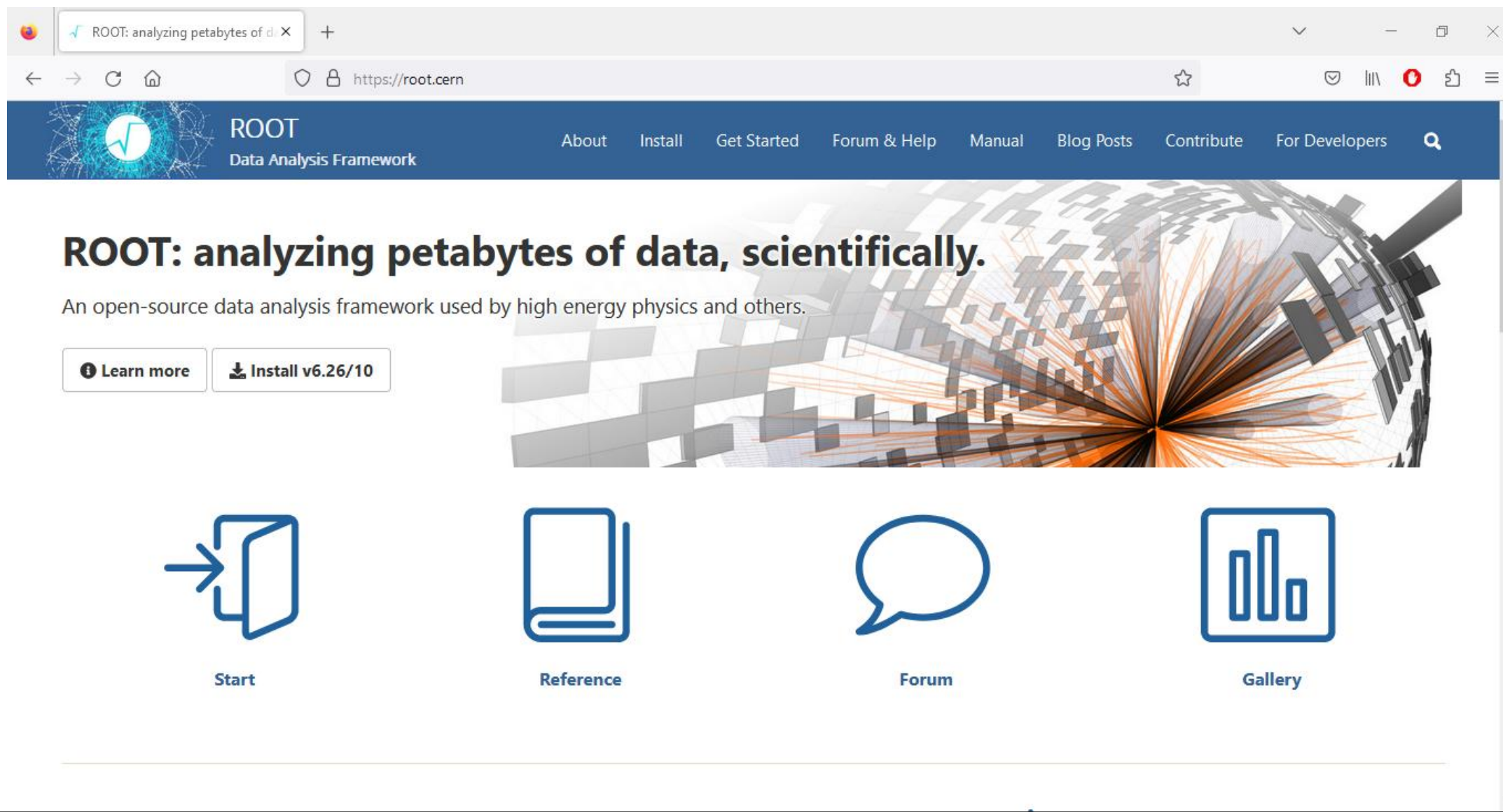
Zobacz "rootlogon.C" na stronie:
github.com/KrzysztofProscinski/ROOT

Makro to służy do wykonywania dodatkowych procesów podczas uruchamiania ROOT-a, np. ładowania dodatkowych bibliotek.

Przy uruchamianiu ROOT-a automatycznie uruchamiane jest makro rootlogon.C, które znajduje się w aktualnie otwartym folderze. Do innych folderów należy utworzyć nowe makra rootlogon.C.

<- W przypadku takiego makra jak to po lewej, przy każdym uruchomieniu ROOT-a napisany zostanie tekst "hello".

Internetowa dokumentacja: root.cern



The image is a screenshot of a web browser displaying the ROOT Data Analysis Framework website. The browser's address bar shows the URL `https://root.cern`. The website's header is dark blue with the ROOT logo on the left and a navigation menu on the right. The main content area features a large banner with the text "ROOT: analyzing petabytes of data, scientifically." and a subtext "An open-source data analysis framework used by high energy physics and others." Below the banner are two buttons: "Learn more" and "Install v6.26/10". At the bottom, there are four icons representing different sections: "Start" (a document with an arrow), "Reference" (a book), "Forum" (a speech bubble), and "Gallery" (a bar chart).

ROOT: analyzing petabytes of data, scientifically.

An open-source data analysis framework used by high energy physics and others.

[Learn more](#) [Install v6.26/10](#)

[Start](#) [Reference](#) [Forum](#) [Gallery](#)

Internetowa dokumentacja: root.cern

The screenshot shows a web browser window displaying the ROOT TH2F Class Reference page. The browser's address bar shows the URL `https://root.cern.ch/doc/master/classTH2F.html`. The page header features the ROOT logo, the text "ROOT Reference Guide", a "Version master" dropdown, and a search bar. A left sidebar lists the "Histogram Library" and "Histogram classes", with "TH2F" selected. The main content area is titled "TH2F Class Reference" and includes a breadcrumb "Histogram Library » Histogram classes.". It describes the "2-D histogram with a float per channel (see TH1 documentation)" and notes its "Definition at line 257 of file TH2.h.". Below this, the "Public Member Functions" section lists several constructors for the TH2F class, each with its signature and a link to "More...". The footer of the page indicates it was generated on Mon Feb 27 2023 09:47:54 (GVA Time) using Doxygen 1.9.5.

ROOT: TH2F Class Reference

Version master

Search

Histogram Library

- Painting classes
- Histogram classes.
 - TAxis
 - TH1
 - TH1C
 - TH1D
 - TH1F
 - TH1I
 - TH1S
 - TH2C
 - TH2D
 - TH2F**
 - TH2I
 - TH2Poly
 - TH2PolyBin
 - TH2S
 - TH3
 - TH3C
 - TH3D
 - TH3F
 - TH3I
 - TH3S

TH2F Class Reference

Histogram Library » Histogram classes.

List of all members | Public Member Functions | Static Public Member Functions | Protected Member Functions | Friends | List of all members

2-D histogram with a float per channel (see TH1 documentation)

Definition at line 257 of file TH2.h.

Public Member Functions

TH2F ()
Constructor. More...

TH2F (const char *name, const char *title, Int_t nbinsx, const Double_t *xbins, Int_t nbinsy, const Double_t *ybins)
Constructor (see TH2::TH2 for explanation of parameters) More...

TH2F (const char *name, const char *title, Int_t nbinsx, const Double_t *xbins, Int_t nbinsy, Double_t ylow, Double_t yup)
Constructor (see TH2::TH2 for explanation of parameters) More...

TH2F (const char *name, const char *title, Int_t nbinsx, const Float_t *xbins, Int_t nbinsy, const Float_t *ybins)
Constructor (see TH2::TH2 for explanation of parameters) More...

TH2F (const char *name, const char *title, Int_t nbinsx, Double_t xlow, Double_t xup, Int_t nbinsy, const Double_t *ybins)
Constructor (see TH2::TH2 for explanation of parameters) More...

TH2F

ROOT master - Reference Guide Generated on Mon Feb 27 2023 09:47:54 (GVA Time) using Doxygen 1.9.5

Proste obliczenia matematyczne

- **co działa, co nie działa**

> 2+3 - dodawanie

> 2*3 - mnożenie

> 2^3 - dodawanie!

<- Poprawny zapis potęgowania to "TMath::Pow(2,3)"

- **TMath**

> TMath::Sqrt(4) - pierwiastek

> TMath::Pi() - liczba pi

> TMath::Sin(0) - sinus

Wszystkie klasy w ROOT-cie zaczynają się od "T".

root.cern.ch/root/html524/TMath.html

<- Na tej stronie znajduje się spis wszystkich funkcji matematycznych w klasie TMath.

Typy zmiennych

Char_t	- char (znak)
Short_t	- short integer (liczba całkowita)
Int_t	- integer (liczba całkowita)
Long64_t	- long64 (liczba całkowita)
Float_t	- float (liczba zmiennoprzecinkowa)
Double_t	- double (float podwójnej precyzji)
Bool_t	- boolean (zmienna boolowska)

Wszystkie typy zmiennych w ROOT-cie kończą się “_t”.

Stałe w ROOT-cie zaczynają się od “k”. Przykładowo klasa Bool_t zawiera dwie stałe: “kTRUE” oraz “kFALSE”.

Niektóre typy zmiennych posiadają wersję “unsigned”, czyli pozbawioną informacji o znaku (+ lub -). Przykładowo Short_t posiada wersję UShort_t. Obie wersje posiadają 16 bitów, więc Short_t obejmuje zakres od -32768 do 32767, natomiast UShort_t obejmuje zakres od 0 do 65535. Istnieją jeszcze UChar_t, UInt_t oraz ULong64_t.

Makra

macro.C

```
"#include <iostream>  
using namespace std;
```

```
Int_t macro(){  
    for(Int_t i=0; i<10; i++){  
        cout<<i<<endl;  
    }  
    return 0;  
}"
```

Zobacz "macro.C" na stronie:
github.com/KrzysztofProscinski/ROOT

A screenshot of a C++ code editor window titled "macro.C" with the path "~/hades/doc". The editor shows the following code:

```
1 #include <iostream>  
2 using namespace std;  
3  
4 Int_t macro(){  
5     for(Int_t i=0; i<10; i++){  
6         cout<<"i"<<endl;  
7     }  
8     return 0;  
9 }
```

The code is color-coded: keywords like "include", "using", "return", and "for" are in red, "Int_t" is in blue, and string literals are in green. The editor has a dark theme and standard window controls (Open, Save, etc.) at the top. The status bar at the bottom indicates "C++", "Tab Width: 8", "Ln 8, Col 1", and "INS" mode.

Makra

- **utworzenie i edycja makra**

> touch macro.C

> nano macro.C

- **uruchamianie**

> root

> .L macro.C //kompilacja (opcjonalna)

> .x macro.C //wykonywanie makra

W ROOT-cie kompilacja nie jest konieczna.
Można wykonać makro od razu.

input, output

- **output**

```
"cout<<"hello"<<endl;"
```

otrzymamy napis hello

```
"Int_t t=0;"
```

```
cout<<t<<endl;"
```

otrzymamy wartość t, czyli 0

- **znaki specjalne**

"\a" - alert/bell

"\n" - newline

"\t" - tab

- **input**

```
"Int_t s;"
```

```
cin >> s;"
```

ustawiamy wartość s na to co wpiszemy z klawiatury

"\b" - backspace

"\r" - return to left margin

Znaki specjalne muszą znajdować się wewnątrz cudzysłowa.

Get, Set

- Set - ustawianie, np.:

```
"TH1F *hist = new TH1F("h1","Title",10,0.,5.);
```

```
hist.SetTitle("nowy tytuł");
```

```
hist.SetMinimum(0);
```

```
hist.SetMaximum(10);"
```

- Get - pobieranie, np.:

```
"TH1F *hist = new TH1F("h1","Title",10,0.,5.);
```

```
hist.GetBinContent(10);
```

```
hist.GetBinError(10);"
```

<- Ustawimy kolejno:

- tytuł histogramu
- dolny histogramu
- górny kres histogramu

<- Otrzymamy wartości kolejno:

- liczby zliczeń w dziesiątym binie histogramu
- niepewności liczby zliczeń w dziesiątym binie histogramu

Obsługa plików

- **TFile**

```
"TFile* f = new TFile ("file.root", "RECREATE");  
TH1F *hist = new TH1F("h1","Title",200,-1.,1.);  
[...]  
f->cd();  
hist->Write();  
f->Close();"
```

- **pobranie obiektu z pliku**

```
"TFile* f = new TFile ("file.root");  
TH1F *hist2= (TH1F*)f -> Get("h1");"
```

Zobacz "read.C" na stronie: github.com/KrzysztofProscinski/ROOT

- **Biblioteki**

<iostream> - input/output na ekran
<fstream> - input do pliku
<ofstream> - output do pliku
<fstream> - input/output do pliku

- **Opcje do wpisania przy definiowaniu pliku:**

CREATE, NEW, READ, RECREATE,
UPDATE

<- W tym przypadku plik file.root musi istnieć już wcześniej i zawierać histogram o nazwie "h1".

Obsługa plików - opcje

Opcja	Krótki opis	Jeśli plik o takiej nazwie wcześniej nie istniał	Jeśli plik o takiej nazwie wcześniej już istniał
CREATE	nowy plik	Utworzony zostaje nowy plik	Stary plik nie zostaje otwarty, a dane nie są nigdzie zapisywane
NEW	nowy plik	Utworzony zostaje nowy plik	Stary plik nie zostaje otwarty, a dane nie są nigdzie zapisywane
READ	plik tylko do odczytu	Nowy plik nie jest tworzony	Stary plik jest otwierany i można pobrać z niego dane, ale nie można go edytować
RECREATE	utworzenie na nowo	Utworzony zostaje nowy plik	Stary plik jest kasowany i zastępowany nowym
UPDATE	wprowadzenie nowych danych do pliku	Utworzony zostaje nowy plik	Nowe dane zostają dodane do już istniejących

Jeżeli przy definicji pliku nie zostanie wpisana opcja, to domyślnie działającą opcją jest "READ".

Opcje "CREATE" i "NEW" działają identycznie.

Histogramy

- **histogram jednowymiarowy**

> TH1F *hist = new TH1F("h1","Histogram",100,0.,10.)

"h1" - nazwa, "Histogram" - wyświetlany tytuł

100 - liczba binów, 0. - dolna krawędź, 10. - górna krawędź

- **histogram dwuwymiarowy**

> TH2F *hist2 = new TH1F("h2","Histogram",100,0.,10.,100,0.,10.);

- **rysowanie**

> hist->Draw();

Histogramy - wypełnianie

- **wypełnianie binów**

```
“for(Int_t i=0; i<1000; i++)  
{hist->Fill(1);}”
```

bin o numerze 1 zostaje wypełniony tysiącem zdarzeń

```
“TRandom3 r;  
r.SetSeed();  
for(Int_t i=0; i<1000; i++)  
{hist->Fill(r.Gaus(0.,1.));}”
```

wypełnienie histogramu tysiącem losowym zdarzeń, wg rozkładu Gaussa

Jest to wypełnianie zdarzenie po zdarzeniu.

Liczba powtórzeń pętli powinna być równa liczbie zdarzeń.

- **ustalanie wartości binów**

```
“for(Int_t i=0; i<100; i++)  
{hist->SetBinContent(i,3);}”
```

każdy bin będzie miał trzy zdarzenia

```
“for(Int_t i=0; i<100; i++)  
{hist->SetBinContent(i, tab[i] );}”
```

każdy bin będzie miał liczbę zdarzeń zgodną z wcześniej przygotowaną tablicą tab[i]

Jest to wypełnianie bin po binie, gdzie wiemy już ile zdarzeń będzie w każdym z binów.

Liczba powtórzeń pętli powinna być równa liczbie binów.

Zobacz “histogram.C” na stronie:
github.com/KrzysztofProscinski/ROOT

Wykresy

- **wykres**

TGraph plot (10,x,y);

10 - liczba rysowanych punktów, x[] - tablica z wartościami x, y[] - tablica z wartościami y

- **przykład**

“Double_t a[] = {1.,2.,3.};

Double_t b[]={0.,1.,0.};

TGraph plot(3,a,b);

plot.Draw();”

- **wykres pobierający dane z pliku**

“TGraph plot(“dane.dat”);”

Zobacz “wykres.C” na stronie:
github.com/KrzysztofProscinski/ROOT

TBrowser

okienkowa przeglądarka (wymagana włączona grafika)

umożliwia otwieranie plików .root i oglądanie zapisanych w nich histogramów oraz wykresów

możliwa jest edycja histogramów (zmiana opcji rysowanie, kolorów, zakresu, liczby binów itd.)

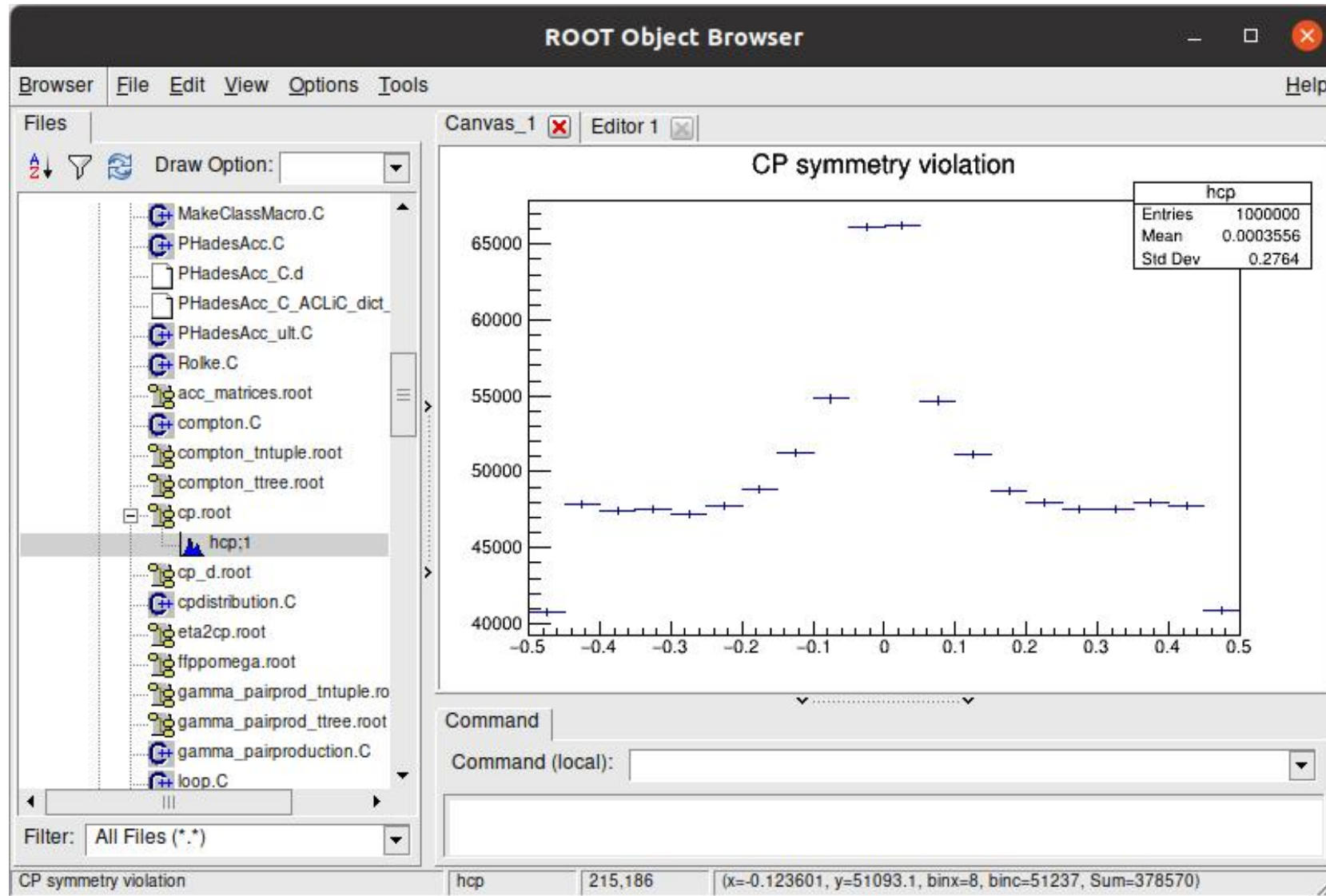
- **uruchamianie**

> root

> new TBrowser

Obsługę TBrowser-a można przećwiczyć na plikach .root
ze strony: github.com/KrzysztofProscinski/ROOT

TBrowser



Część 2

TBrowser

- File -> Open
 - otwarcie nowego pliku
- File -> Save
 - zapis wyświetlanego histogramu
- Edit -> Clear -> Pad
 - wyczyszczenie okna
- View -> Editor
 - panel edycji
- View -> Toolbar
 - panel narzędzi (do dodawania nowych obiektów)
- View -> Event Statusbar
 - informacja o wskazywanym myszką binie
- Tools -> Fit Panel
 - panel fitowania
- Tools -> Event Recorder
 - odtwarzanie zdarzeń w kolejności rejestracji
(dla plików posiadających takie informacje)

Files Pad Editor

Style

Name

TEllipse::TEllipse

Line

2

1

Opacity

Fill

1

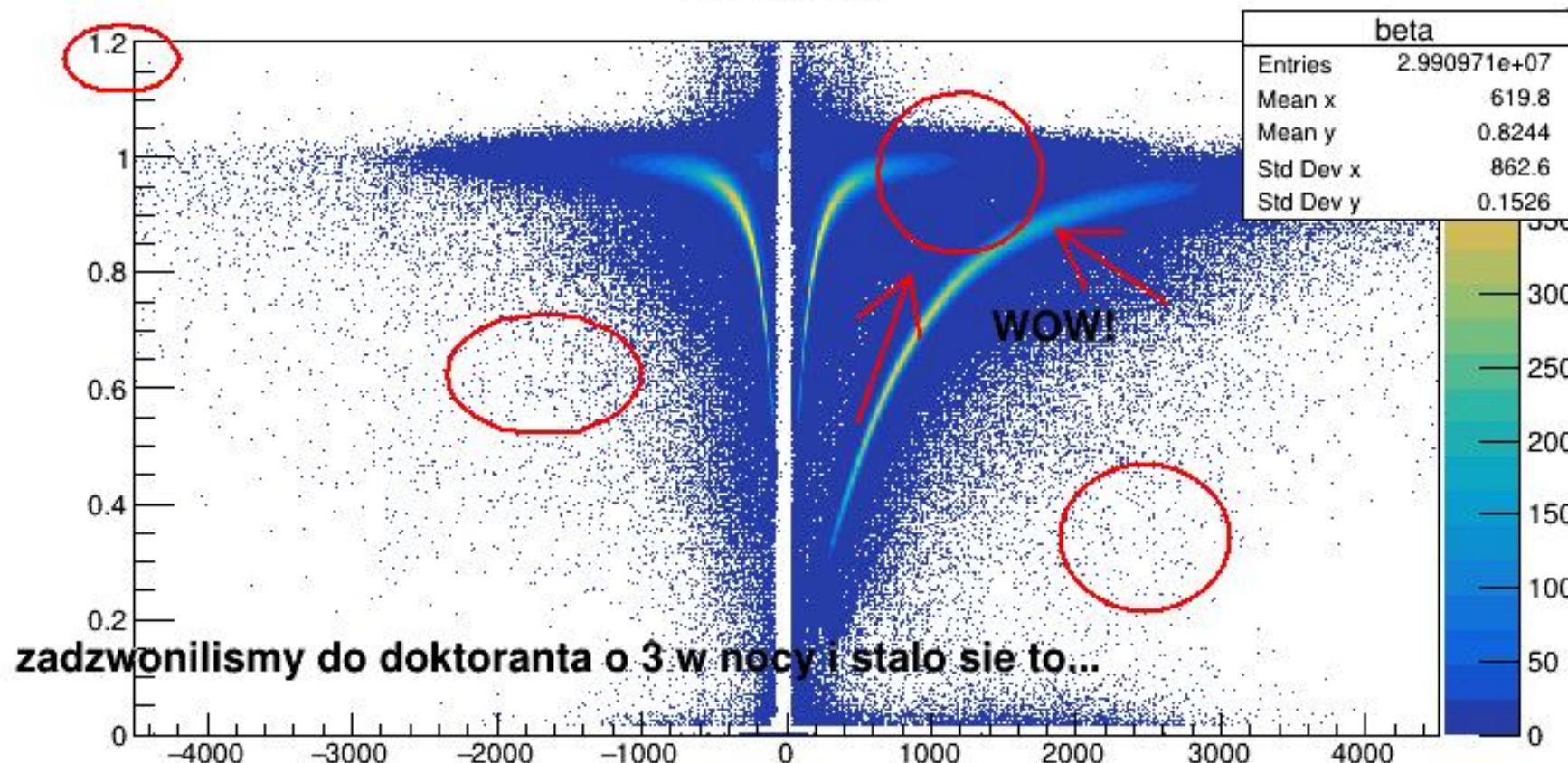
Opacity

1

Canvas_1 Editor 1



beta vs mom



Czy wiesz, że używając panela narzędzi i narzędzia edycji można tworzyć miniaturki na YouTube'a?

Command

Command (local):

Obiekty graficzne

- **Obiekty:**

TMarker	- punkt
TLine	- linia
TArrow	- strzałka
TBox	- kwadrat
TEllipse	- elipsa
TText	- tekst
TLatex	- tekst LaTeX

- **Opcje:**

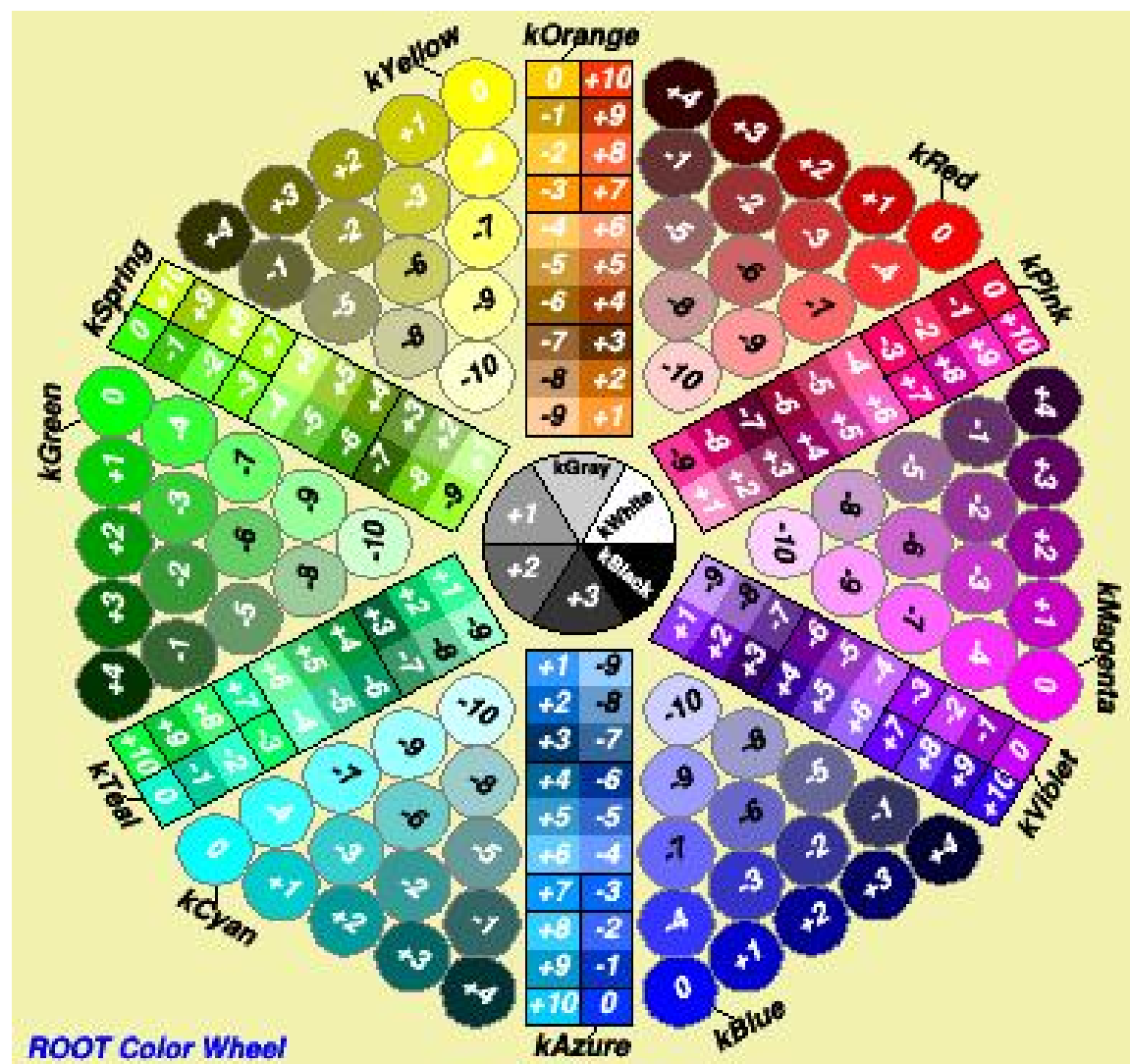
SetMarkerColor(1)	- kolor punktu
SetMarkerSize(1.0)	- rozmiar punktu
SetFillColor(kRed+1)	- kolor kwadratu
SetFillStyle(3002)	- styl wypełnienia
SetTextFont(40)	- rozmiar tekstu
SetTextAngle(45)	- kąt tekstu
Draw()	- rysowanie
DrawLatex(0.5,0.6,"E^{2}")	- LaTeX

Dokładniejsze informacje znajdują się w poradniku
dr-a Piaseckiego: www.fuw.edu.pl/~kpias/

Kolory

40	41	42	43	44	45	46	47	48	49
30	31	32	33	34	35	36	37	38	39
20	21	22	23	24	25	26	27	28	29
10	11	12	13	14	15	16	17	18	19
0	1	2	3	4	5	6	7	8	9

Kolory



W ROOT-cie istnieją dwa zestawy kolorów, zaprezentowane na tym i poprzednim slajdzie.

Przykładowo żeby narysować histogram o nazwie "hist" w kolorze bardzo jasnym czerwonym należy wpisać `hist->SetLineColor("kRed-10")`.

Histogramy - opcje rysowania

• Histogramy 1D

c - dodanie krzywej łączącej biny

e - dodanie błędów

hist - biny, bez błędów

l - dodanie linii łączącej biny

lego - histogram LEGO

pie - "pie chart"

surf - powierzchnia

same - nałożenie na poprzedni histogram

**Opcje można ze sobą łączyć
(ale niektóre wzajemnie się wykluczają)**

• Histogramy 2D

arrow - histogram z wektorów

box - histogram z "box-ów"

colz - kolorowy histogram

cont - wykres konturowy

lego - histogram LEGO

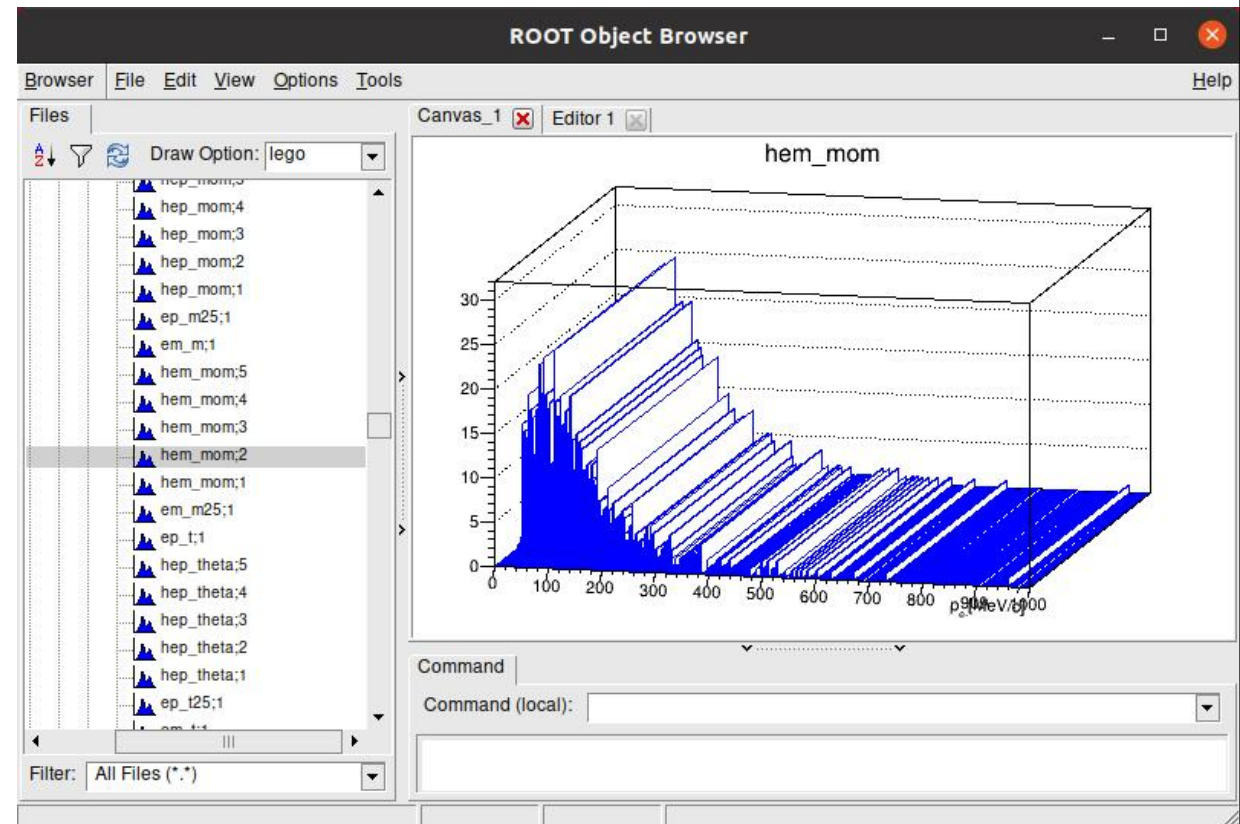
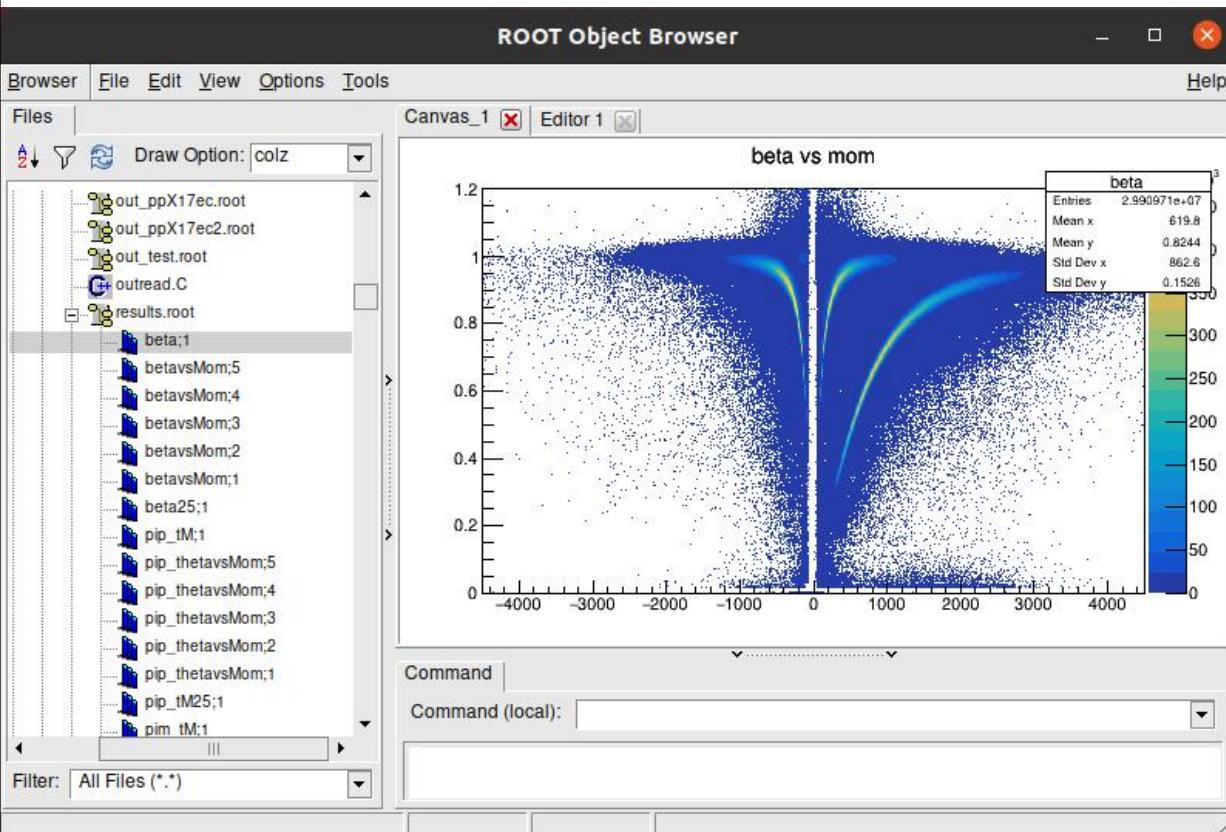
surf - powierzchnia

text - histogram z liczb

same - nałożenie na poprzedni histogram

Opcje należy wpisać przy rysowaniu histogramu, czyli np. hist->Draw("e").

Histogramy - opcje rysowania



Przykłady opcji "colz" dla histogramu dwuwymiarowego i opcji "lego" dla histogramu jednowymiarowego.

Legenda

```
"TLegend* legenda = new TLegend(0.1,0.6,0.48,0.9);  
legenda->SetHeader("Tytuł legendy");  
legenda->AddEntry(hist1,"Histogram 1");  
legenda->AddEntry(hist2,"Histogram 2","l");  
legenda->Draw();"
```

- Opcje rysowanie

- e - dodanie błędu
- f - wypełnienie pod histogramem
- l - linia
- p - punktowy znacznik

Argumenty w TLegend to współrzędne lewego dolnego rogu i prawego górnego rogu rysowanej legendy.

Zobacz "grafika_i_legenda.C" na stronie:
github.com/KrzysztofProscinski/ROOT

Tl_{at}ex

- **Składnia w makrze:**

```
"Tlatex I;
```

```
I.SetTextSize(4);
```

```
I.SetTextAngle(0.);
```

```
I.SetTextColor(1);
```

```
I.DrawLatex(0.5,0.6,"x2");"
```

Tl_{at}ex służy do wstawiania tekstu pisanego zgodnie z notacją LaTeX_a, tylko z paroma różnicami.

różnice	w Overleaf-ie	w ROOT-cie
znak	<code>\alpha</code>	<code>#alpha</code>
funkcje	<code>\frac{ }{ }</code>	<code>#frac{ }{ }</code>
indeks górny	<code>^x</code>	<code>^{x}</code>
indeks dolny	<code>_x</code>	<code>_{x}</code>

Funkcje

- **Składnia w makrze:**

`"TF1 f1 ("f1","x*x",0.,10.);`

`f1.SetRange(-10.,10.);`

`f1.Eval(3.);`

`f1.Integral(0.,5.);`

`f1.GetMinimum(0.,5.);`

`f1.SetLineColor(2);`

`f1.SetLineStyle(9)`

`f1.Draw();"`

definicja (możliwe też TF2 i TF3)

zakres funkcji

wartość

całka oznaczona dla podanych granic

minimum w podanym przydziale

kolor linii

styl linii

rysowanie

Zobacz "funkcja.C" na stronie:
github.com/KrzysztofProscinski/ROOT

Funkcje służą do tworzenia własnych funkcji matematycznych. Argumenty TF1 to nazwa funkcji, postać (w powyższym przypadku jest to $f(x)=x^2$), dolna krawędź i górna krawędź zakresu funkcji.

Fitowanie funkcji

- **Definicja**

```
"Double_t MyFunction(Double_t *arg, Double_t *par){  
    Double_t f = par[0]+par[1]*arg[0]+par[2]*arg[0]*arg[0];  
    return f;}"
```

- **Fitowanie**

```
"TF1 *f1 = new TF1("funkcja",MyFunction,0,100,3);  
f1->SetParameters(0.,-2.,1.);  
TFitResultPtr results = hist1->Fit(f1,"S");  
cout << "Chi2 = " << results->Chi2() << endl;"
```

Zobacz "fitowanie.C" na stronie:
github.com/KrzysztofProscinski/ROOT

Definicje funkcji trzeba wprowadzić poza główną częścią makra. Jeżeli np. mamy makro fitowanie.C, to powinno ono wyglądać tak:

```
"Double_t MyFunction([...]){  
    [...]  
}  
Double_t fitowanie([...]){  
    [...]  
}."
```

3 w definicji TF1 oznacza liczbę parametrów funkcji do fitowania. W tym przypadku mamy parametry par[0], par[1] i par[2], czyli są 3.

Fitowanie funkcji

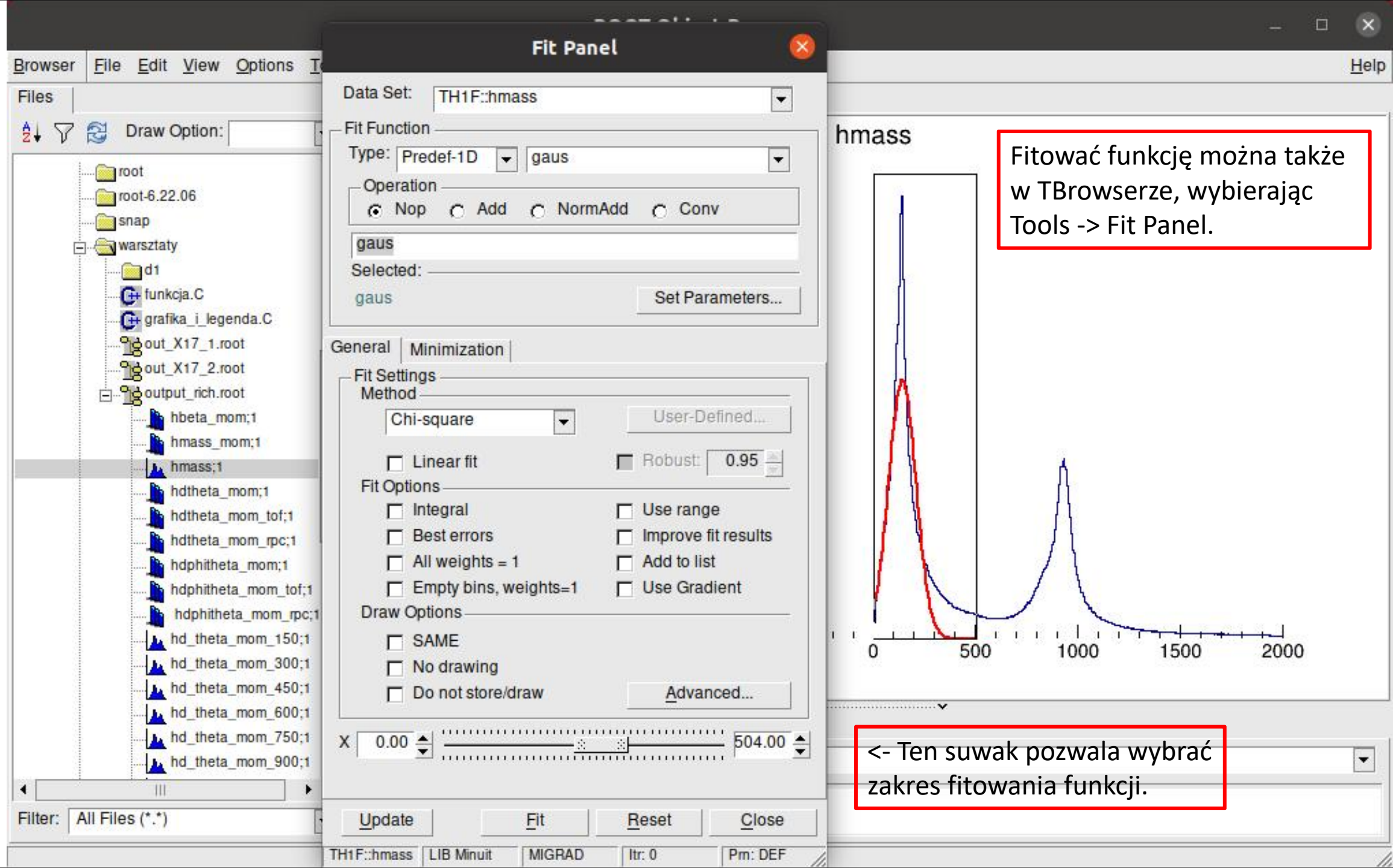
- **Opcje fitowania:**

- I - całkowanie binu zamiast brania wartości w centrum
- L - fit metodą "log-likelihood"
- M - poprawa fitu przez algorytm TMinuit
- P - fit metodą "Pearson chi-square"
- R - zakres fitu zgodny z zakresem funkcji
- S - do rezultatu fitu można się odnieść (klasa TFitResultPtr)
- W - waga każdego binu równa 1 (bez względu na niepewości)
- WL - fit metodą "weighted log likelihood"
- WW - waga każdego binu równa 1, wliczając puste biny
- 0 - bez rysowania dofitowanej krzywej

- **Predefiniowane funkcje:**

exp
gaus
landau
poin

Predefiniowanych funkcji nie trzeba definiować jeszcze raz. Wystarczy napisać np.:
"TF1 f1 ("f1",gaus,0.,10.)".



Generator liczb losowych

- **Składnia**

> TRandom3 rand;

> rand.SetSeed();

> rand.Rndm();

- **Wypełnianie histogramów**

```
"TH1F* h = new TH1F("h","",50,-5.,5.);
```

```
f->FillRandom("gaus",10000);"
```

- **Rozkłady**

Binomial(intot,prob)

BreitWigner(mean,gamma)

Exp(tau)

Gaus(mean,sigma)

Integer(imax)

Landau(mean,sigma)

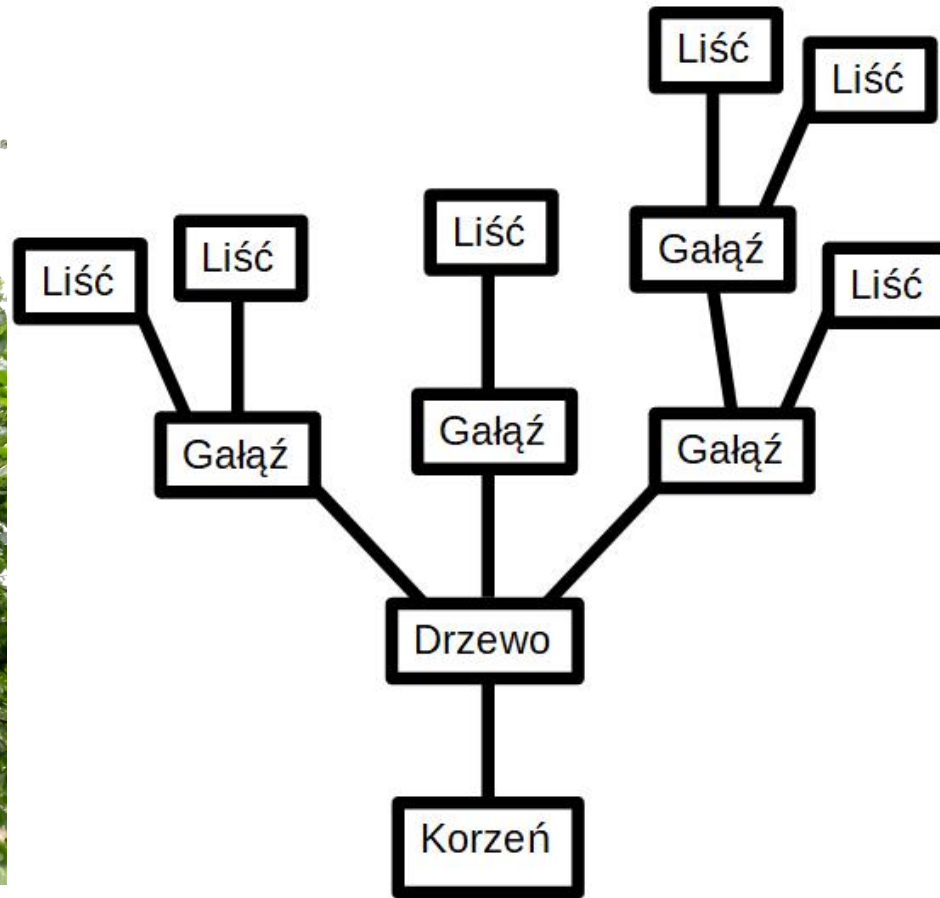
Poisson(mean)

Rndm()

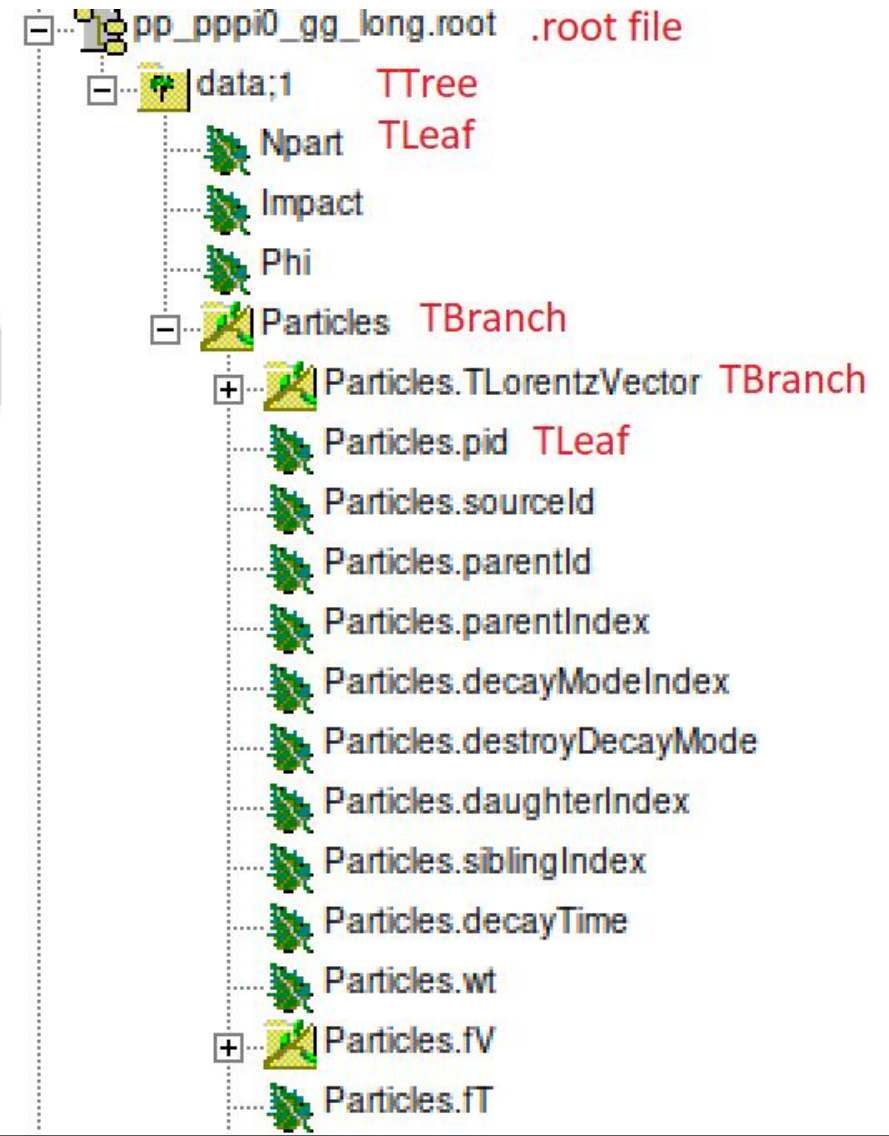
Dostępne rozkłady
liczb losowych:

- dwumianowy
- Breita-Wignera
- eksponencjalny
- Gaussa
- jednorodny
- Landau'a
- Poissona
- jednorodny w przedziale (0,1)

TTree



Każdy liść drzewa zawiera wartości jednego parametru. Każde zdarzenie ma po jednej wartości w każdym z liści.



TTree

- **Definicja**

```
“Float_t energy, momentum;  
  Int_t charge;  
  TFile f ("tree.root","RECREATE");  
  TTree* t = new TTree("tree","Tree 1");  
  t->Branch("E",&energy,"Energy/F");  
  t->Branch("mom",&momentum,"Momentum/F");  
  t->Branch("charge",&charge,"Charge/B");“
```

- **Zmienne:**

O - bool

D - double

F - float

B - integer (1 bajt)

S - integer (2 bajty)

I - integer (4 bajty)

F - integer (8 bajtów)

Zobacz “drzewo.C” na stronie:
github.com/KrzysztofProscinski/ROOT

TTree

- **Wypełnianie**

```
"TRandom3 r;  
r.SetSeed();  
for(Int_t i=0; i<1000; i++){  
    energy = r.Gaus();  
    momentum = r.Gaus();  
    charge = 1;  
    t->Fill();  
}  
t->Write();"
```

- **Obsługa**

```
>t.ReadFile("t1.txt","energy/F:momentum/F:  
charge/B")
```

Zapis danych z pliku.

```
> t.Print()
```

Informacje o drzewie.

```
> t.Show(10)
```

Parametry 10. zdarzenia.

```
> t.Scan("energy:momentum:charge")
```

Parametry każdego zdarzenia.

```
> t.Draw("energy")
```

Histogram wybranego parametru.

TNtuple (drzewo wypełniane Float-ami)

- **Definicja i wypełnianie**

```
"Float_t energy, momentum, charge;  
TFile f ("ntuple.root","RECREATE");  
TNtuple Nt ("ntuple","N tuple 1","Energy:Momentum:Charge");  
TRandom3 r;r.SetSeed();  
for(Int_t i=0; i<1000; i++){  
    energy = r.Gaus();  
    momentum = r.Gaus();  
    charge=1;  
    Nt.Fill(energy,momentum,charge);  
}"
```

Jeżeli parametrów jest mało,
to można poprzestać na
gałęziach, bez dzielenia ich na
osobne liście.

Zobacz "Ntuple.C" na stronie:
github.com/KrzysztofProscinski/ROOT

Wektory

- **Definicje**

> TVector2 v0 (1,-1) Wektor 2D.

> TVector3 v1 (1,2,0) Wektor 3D.

- **Operacje**

> v1.SetXYZ (2,1,0) Zmiana współrzędnych.

> 2*v1 Mnożenie przez skalar.

> v1+v2 Dodawanie wektorów.

> v1.Dot(v2) Iloczyn skalarny.

> v1.Cross(v2) Iloczyn wektorowy.

> v1.Rotate(0,0,TMath::Pi()) Obrót.

- **Parametry**

> v1.Mag() Długość.

> v1.Mag2() Kwadrat długości.

> v1.Theta() Kąt azymutalny.

> v1.CosTheta() Cosinus kąta theta.

> v1.Phi() //kąt aksjalny Kąt aksjalny.

> v1.Orthogonal() Wektor prostopadły do v1.

> v1.Angle(v2) Kąt między wektorami.

4-wektory

- **Definicja**

> TLorentzVector lv (1,0,0,1);

- **Struktura**

> (x,y,z,t) lub (px,py,pz,E)

- **Operacje**

> lv.Boost(v1)

Boost w kierunku
określonym wektorem v1.

- **Parametry**

> lv.M()

> lv.P()

> lv.Pt()

> lv.Beta()

> lv.Gamma()

> lv.Rapidity()

Masa.

Pęd.

Pęd transwersalny.

Relatywistyczny czynnik beta.

Relatywistyczny czynnik gamma.

Pospieszność.

Praca na listach

- **makro do list**

```
"char name[10], title[20];
TObjArray Hlist(0);
TH1F* h;
for (Int_t i = 0; i < 15; i++) {
    sprintf(name,"h%d",i);
    sprintf(title,"histo nr:%d",i);
    h = new TH1F(name,title,100,-4,4);
    Hlist.Add(h);
    h->FillRandom("gaus",1000);
}"
```

Zobacz "lista.C" na stronie: github.com/KrzysztofProscinski/ROOT

- **canvas**

```
"TCanvas* canv = new TCanvas([...]);
canv->Divide(4,4);
for (Int_t i = 1; i < 16; i++) {
    canv->cd(i);
    Hlist[i]->Draw();
    canv->Update();
}"
```

Canvas to okno na którym rysowane są histogramy. Normalnie canvas tworzony jest automatycznie, ale można zdefiniować go samemu.

- **tworzenie list w terminalu**

```
> ls *.root >> lista_plikow.list
```

Aby stworzyć listę w ten sposób należy wyjść z ROOT-a.

TChain, hadd

- **TChain**

```
"TChain chain1 ("tree");  
chain1.Add ("data1.root");  
chain1.Add ("data2.root");  
chain1.Add ("data3.root");  
[...]  
chain1.Draw();"
```

- **hadd**

```
> hadd sum.root file_*.root
```

Aby dodawać pliki
w ten sposób należy
wyjść z ROOT-a.

pliki do połączenia:

file_1.root

file_2.root

plik, który chcemy stworzyć:

sum.root

Dodawanie histogramów

- **Wykonywanie**

```
"TFile *f1 = new TFile("out1.root");  
TH1F *hcp1 = (TH1F*)f1->Get("hsincos");  
TFile *f2 = new TFile("out1.root");  
TH1F *hcp2 = (TH1F*)f1->Get("hsincos");  
TH1F *hcp_sum = new TH1F("sincos","sin cos",200,-1.,1.);  
hcp_sum->Add(hcp1,hcp2,1,1);  
hcp_sum->Draw();"
```

Argumenty w funkcji Add to:
pierwszy histogram, drugi
histogram, waga z jaką jest
dodawany pierwszy
histogram, waga z jaką jest
dodawany drugi histogram.