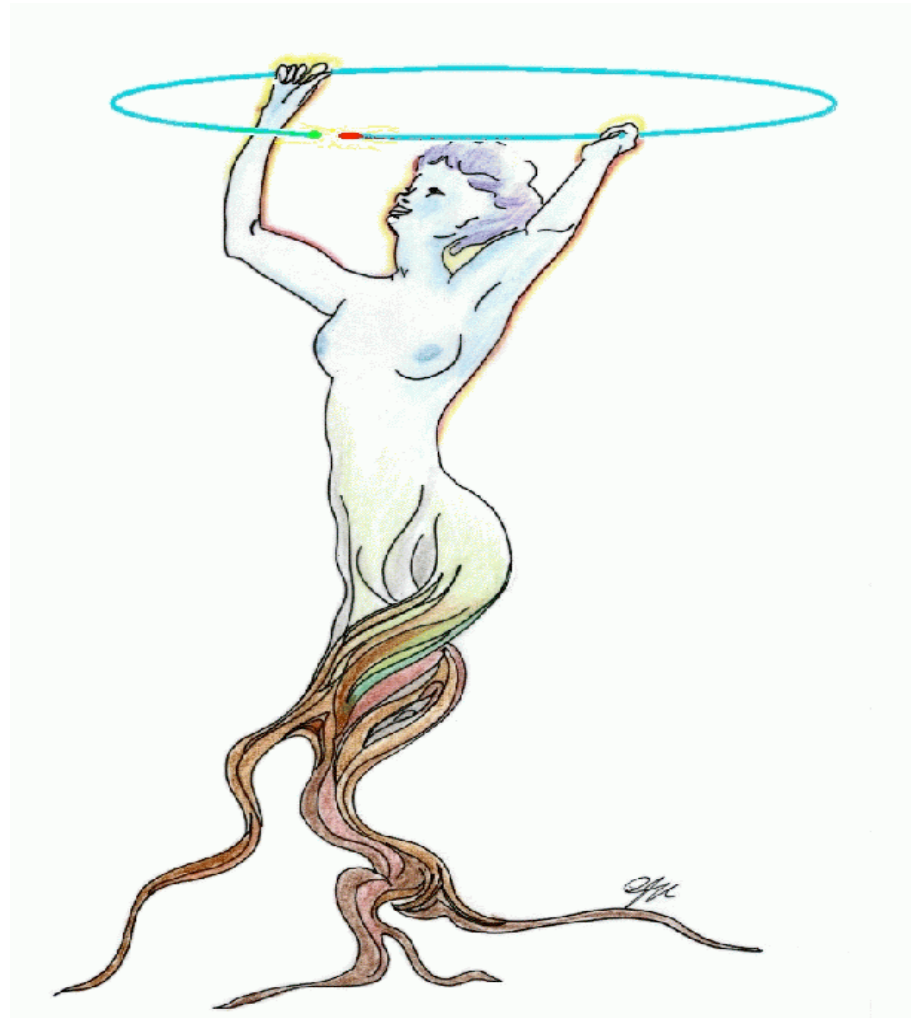


# *ROOT for beginners*

*Fourth day*

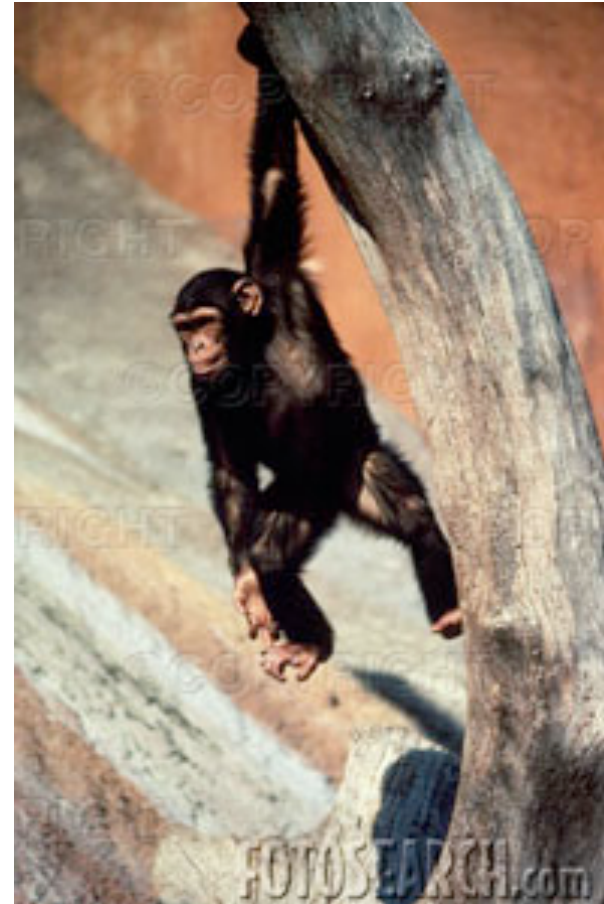
Trees



# *Let us climb on trees...*

## **Today we will:**

- Create a tree
- Fill it
- Read it
- Make analyses
- ...



*Create a tree*

# *In the shadow of my tree...*

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

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

tree title

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

tree branches contain the variables (leaves)

name of the branch

Name and type of the variable

variable address in the memory

# Defining the branches

- Simple variables

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

- Fixed size array

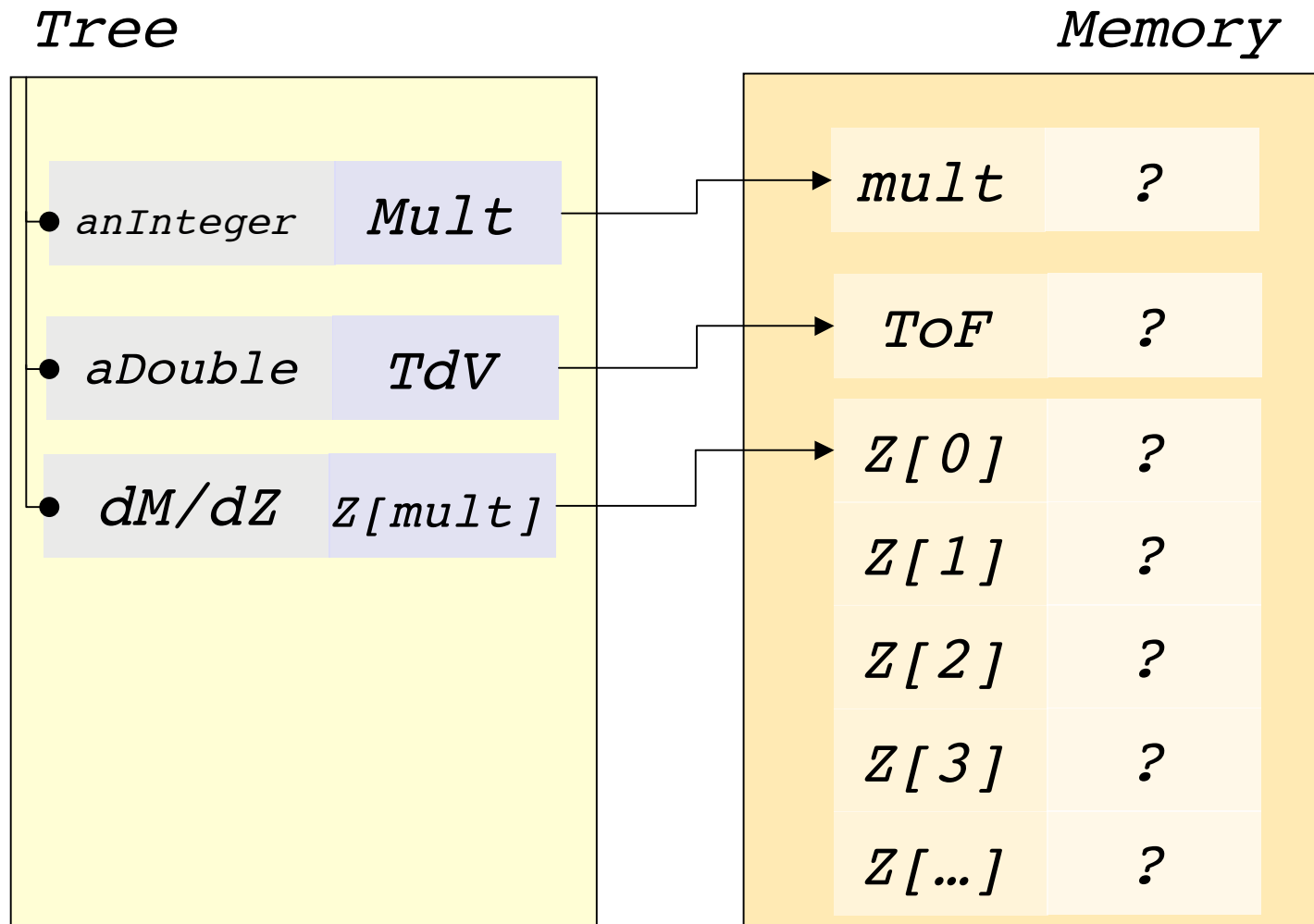
```
Double_t z[50];  
tree->Branch("z_branch", z, "Charge[50]/D");
```

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

- Variable size array

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

# What happens in memory...

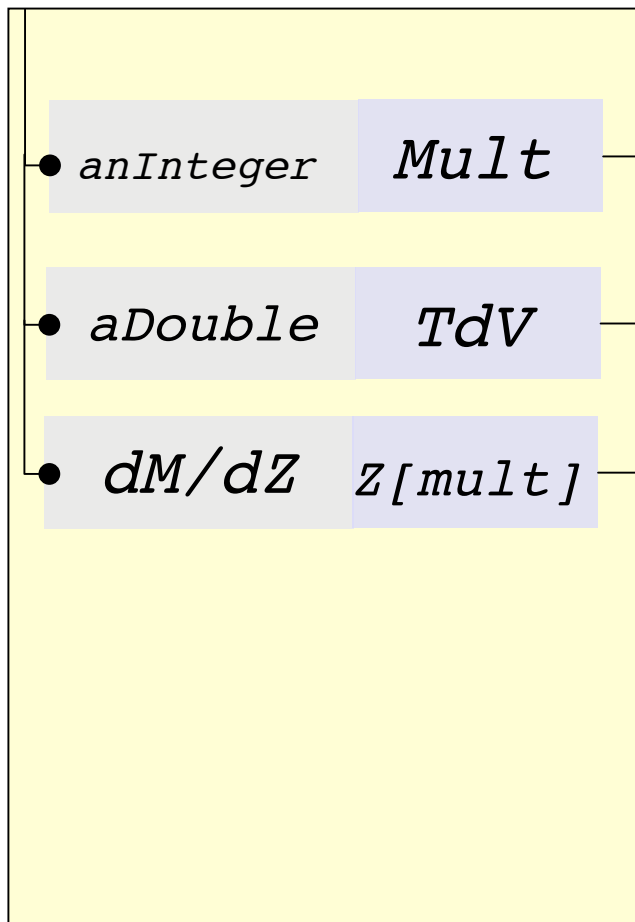


# What happens in memory...

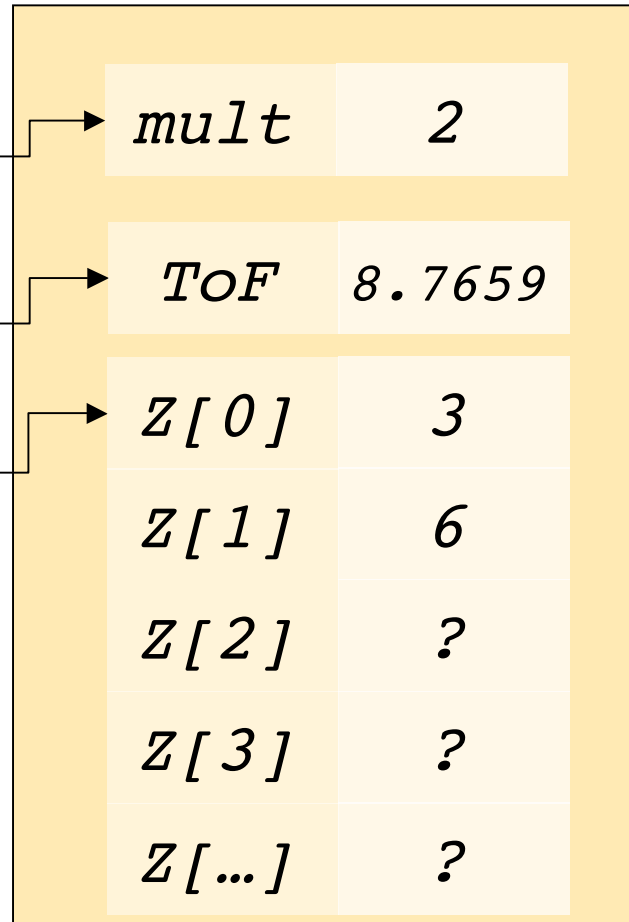
*Writing to the file*

<code>mult=2</code>	<code>Z[0]=3</code>
<code>ToF=8.7659</code>	<code>Z[1]=6</code>

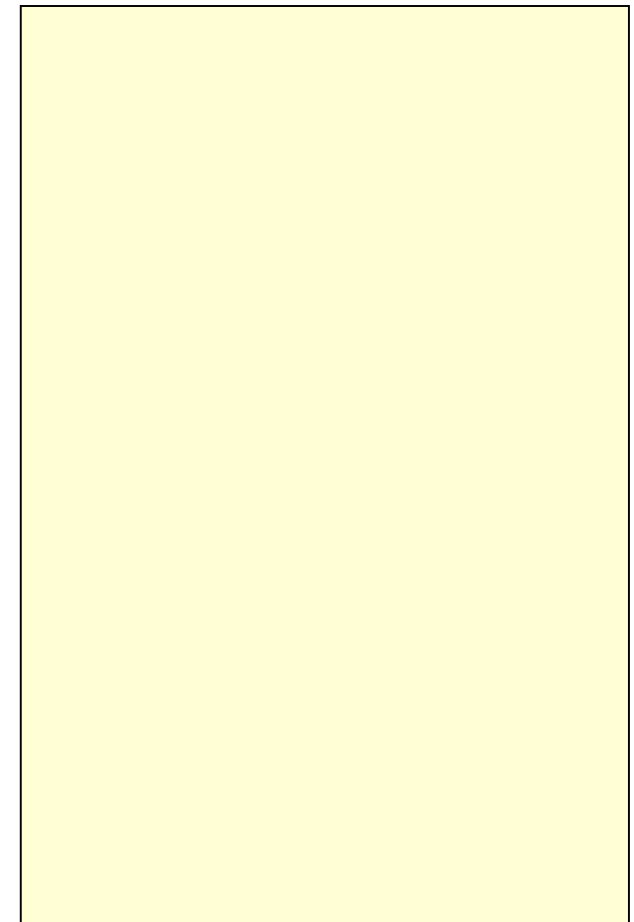
*Tree*



*Memory*



*File*

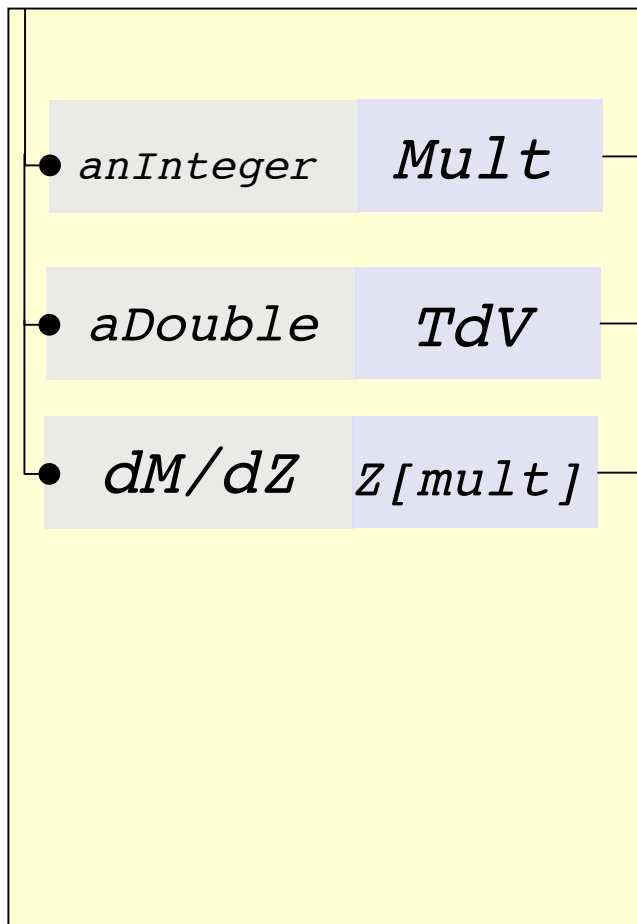


# What happens in memory...

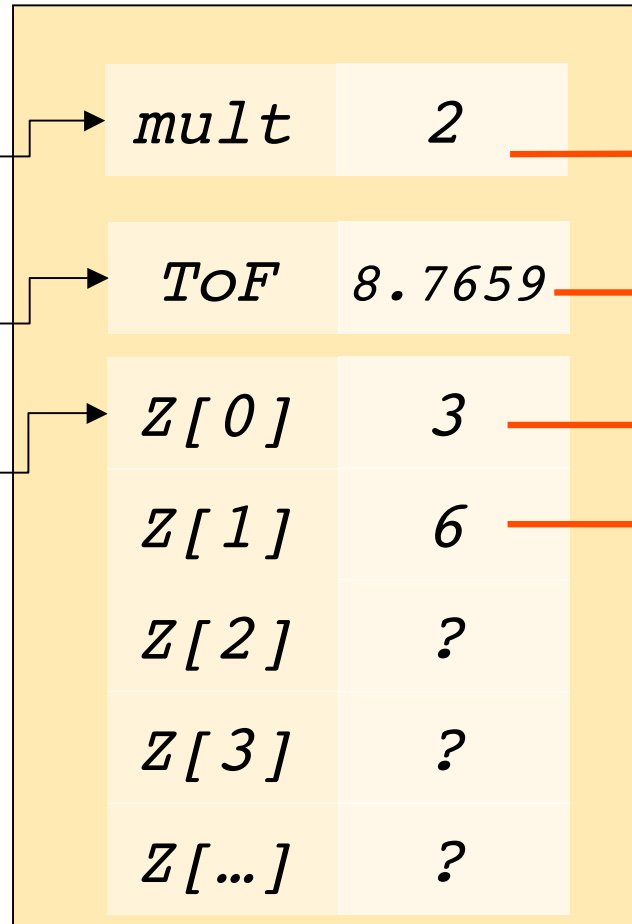
Writing to the file

```
tree->Fill()
```

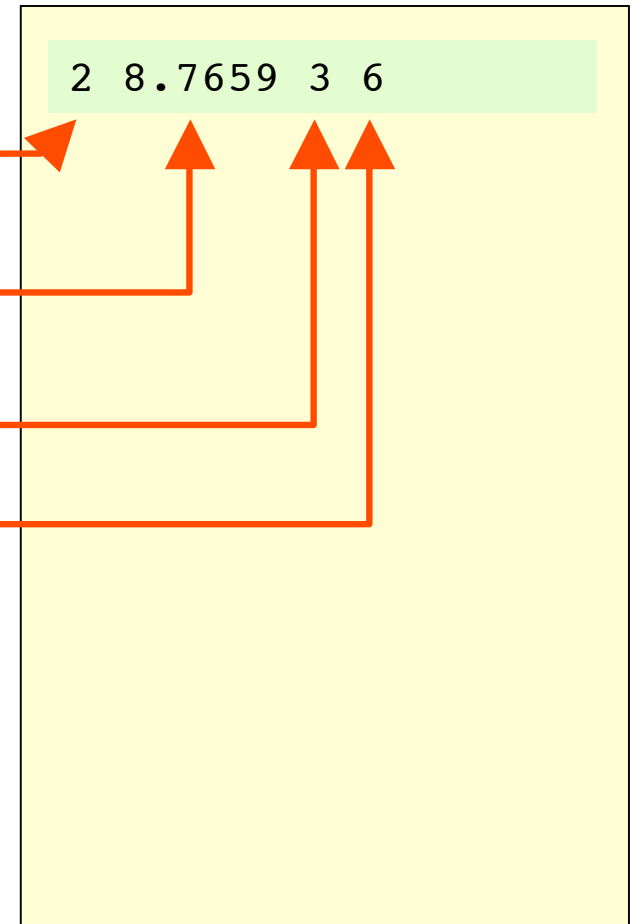
Tree



Memory



File



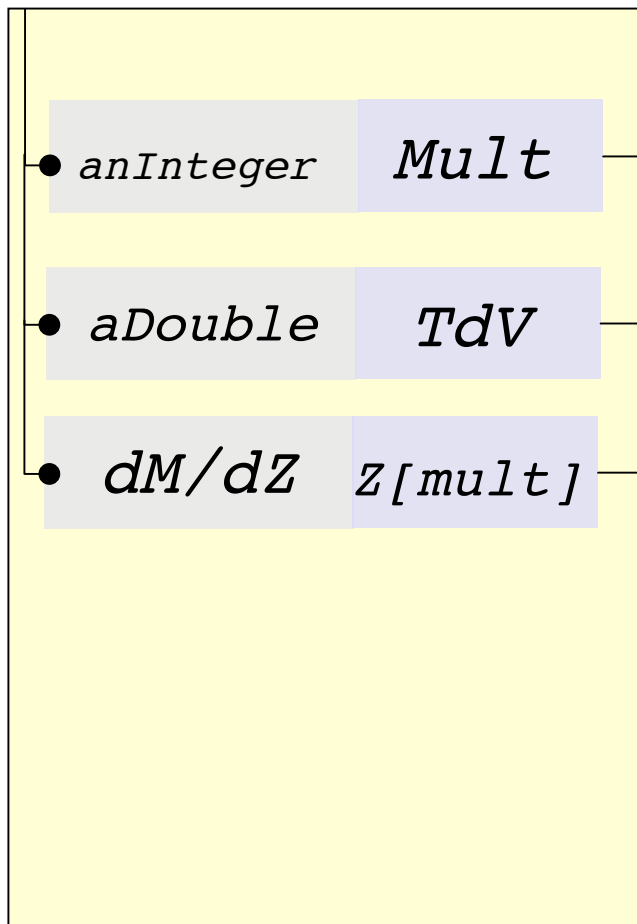


# What happens in memory...

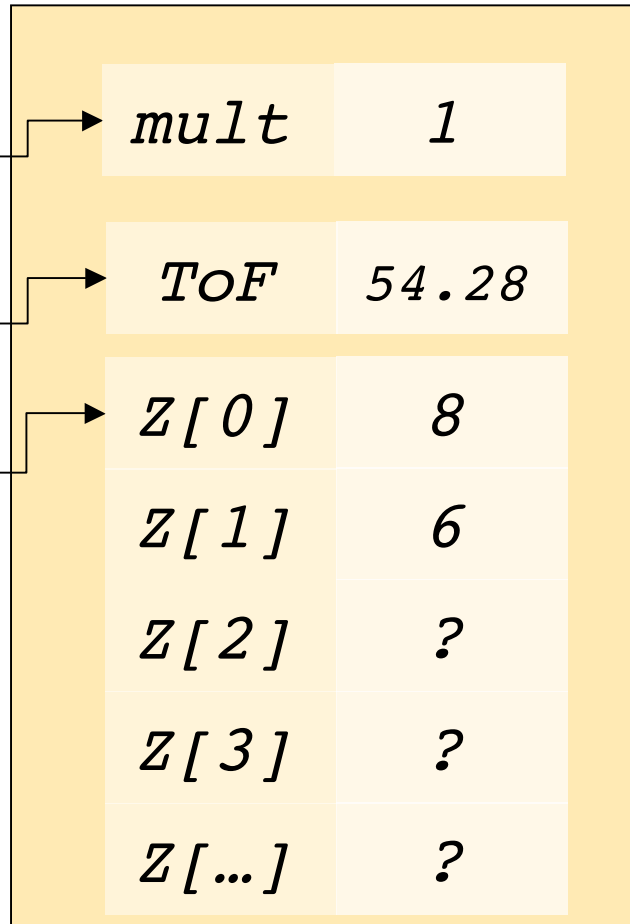
*Writing to the file*

```
mult=1  
ToF=54.28  
Z[0]=8
```

*Tree*



*Memory*



*File*

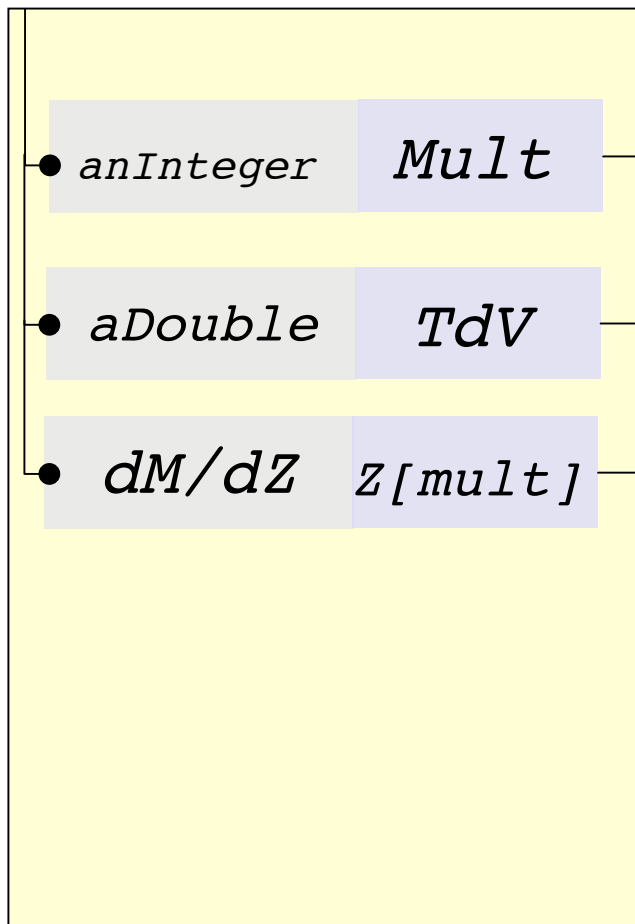
```
2 8.7659 3 6
```

# What happens in memory...

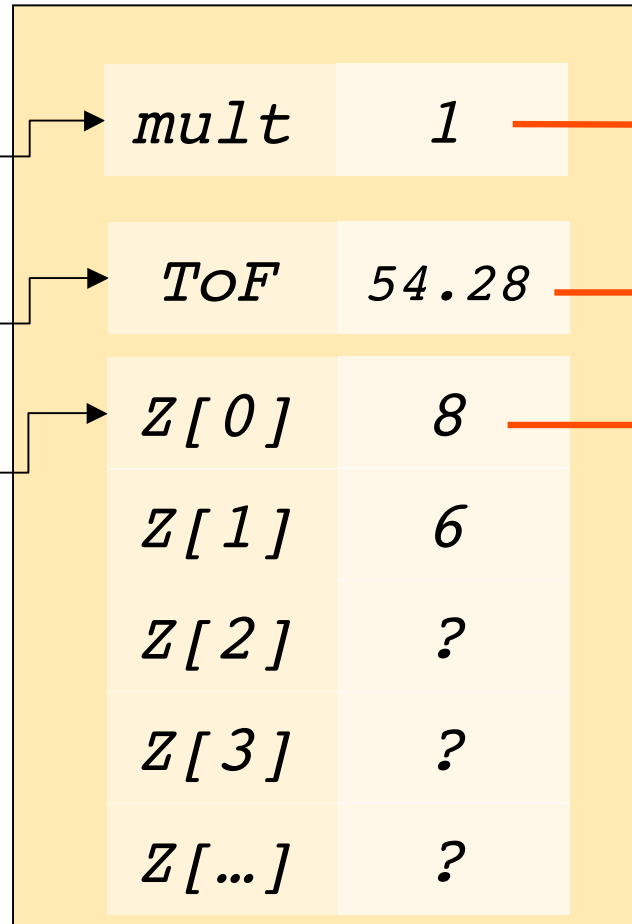
Writing to the file

```
tree->Fill()
```

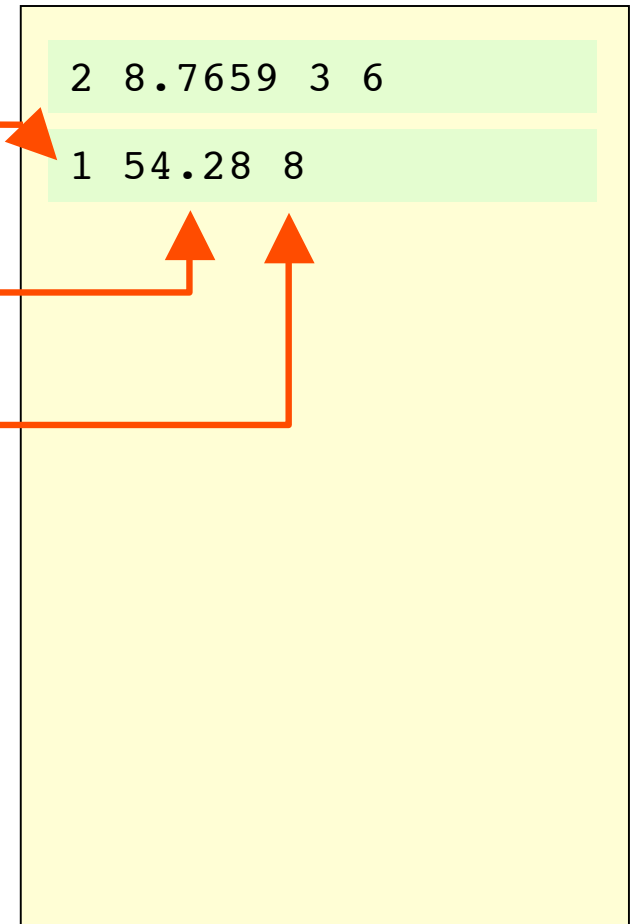
Tree



Memory



File

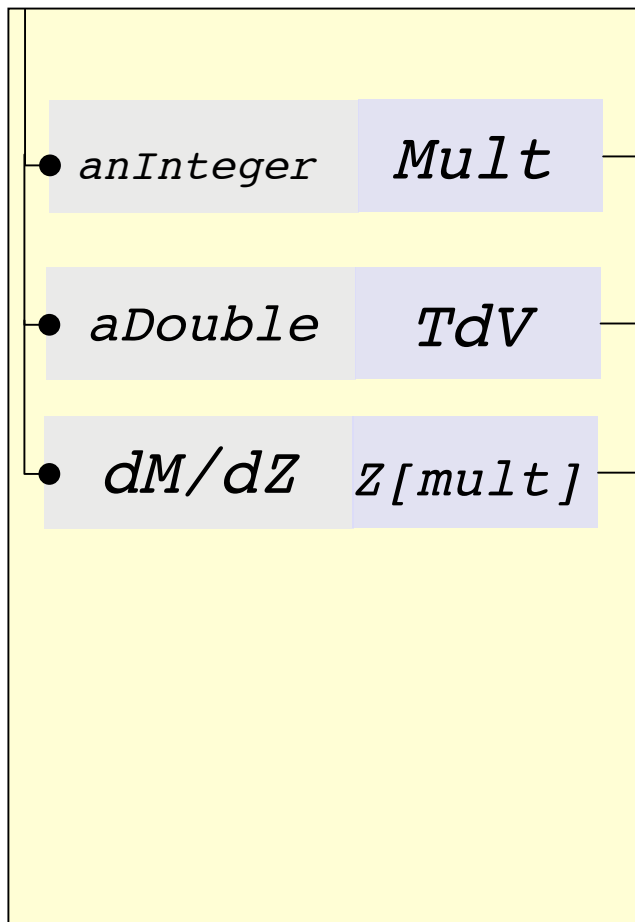


# What happens in memory...

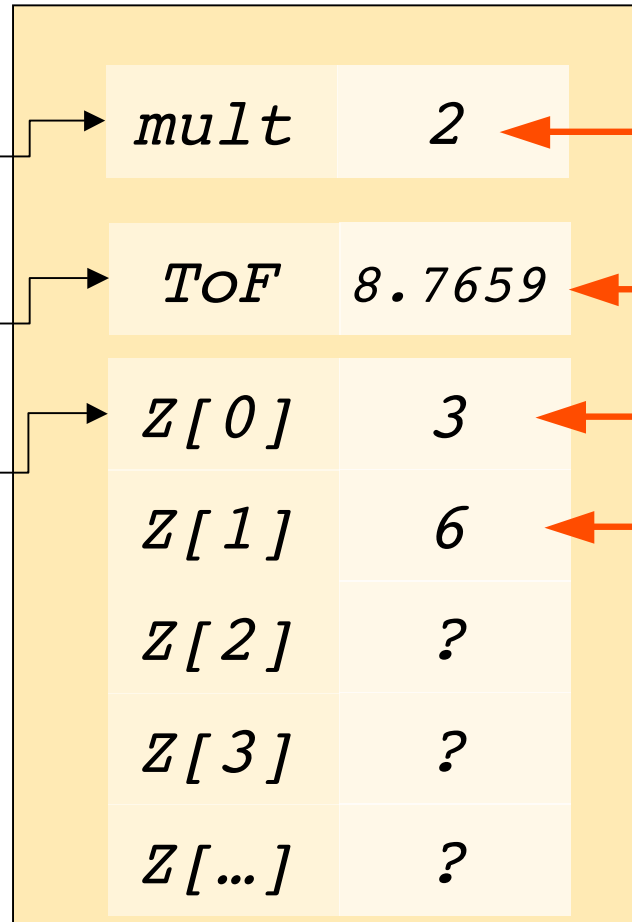
*Reading the file*

```
tree->GetEntry(0)
```

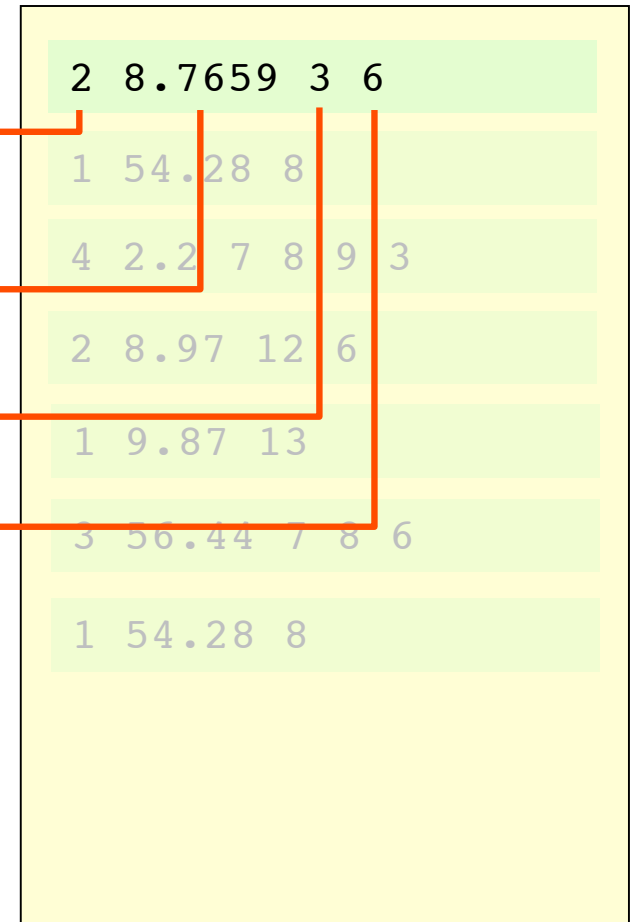
*Tree*



*Memory*



*File*

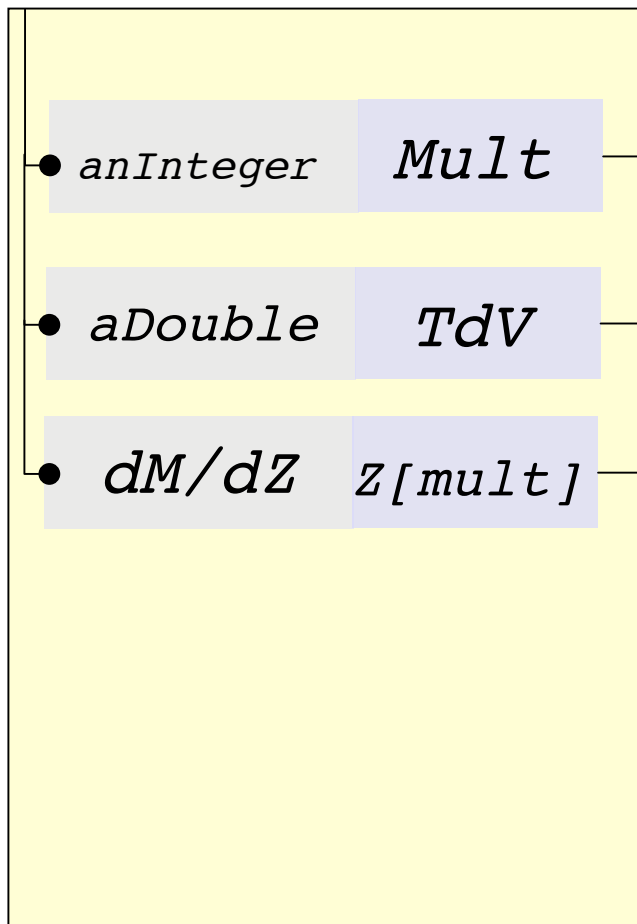


# What happens in memory...

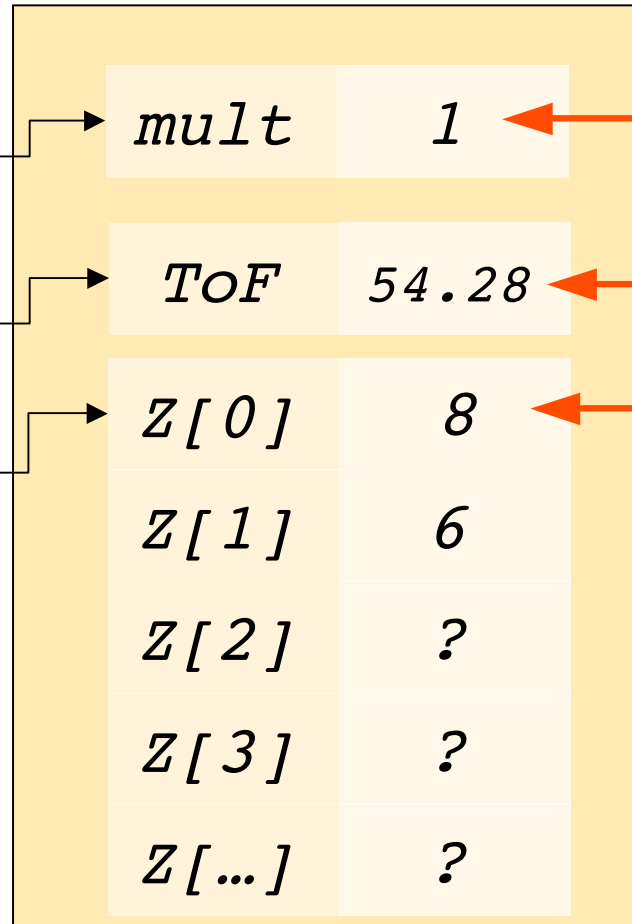
*Reading the file*

```
tree->GetEntry(1)
```

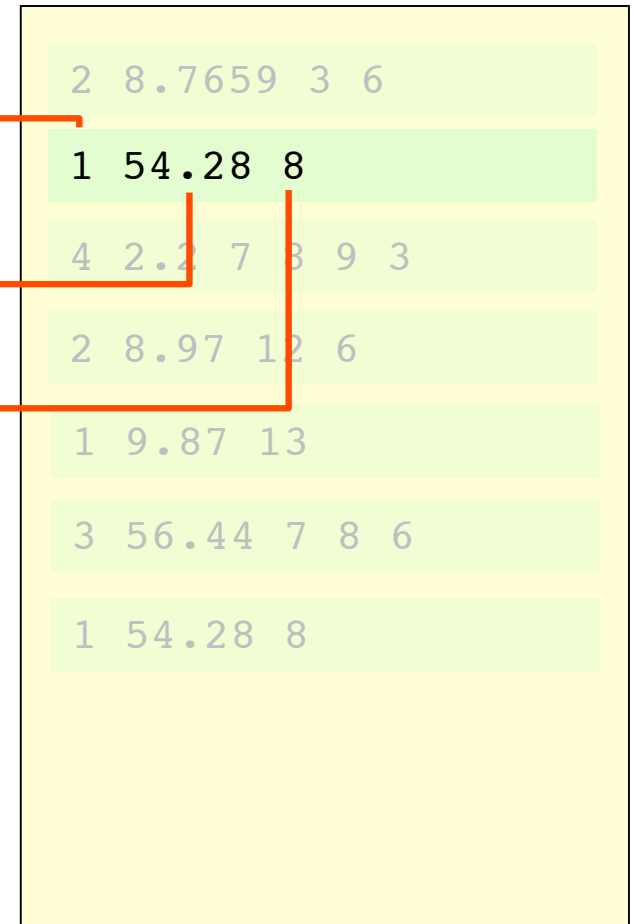
*Tree*



*Memory*



*File*

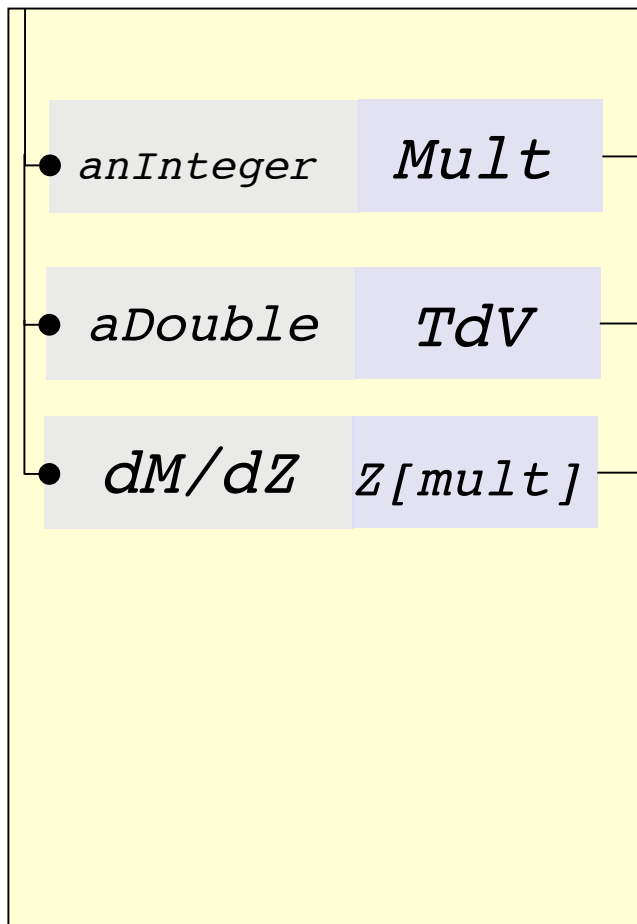


# What happens in memory...

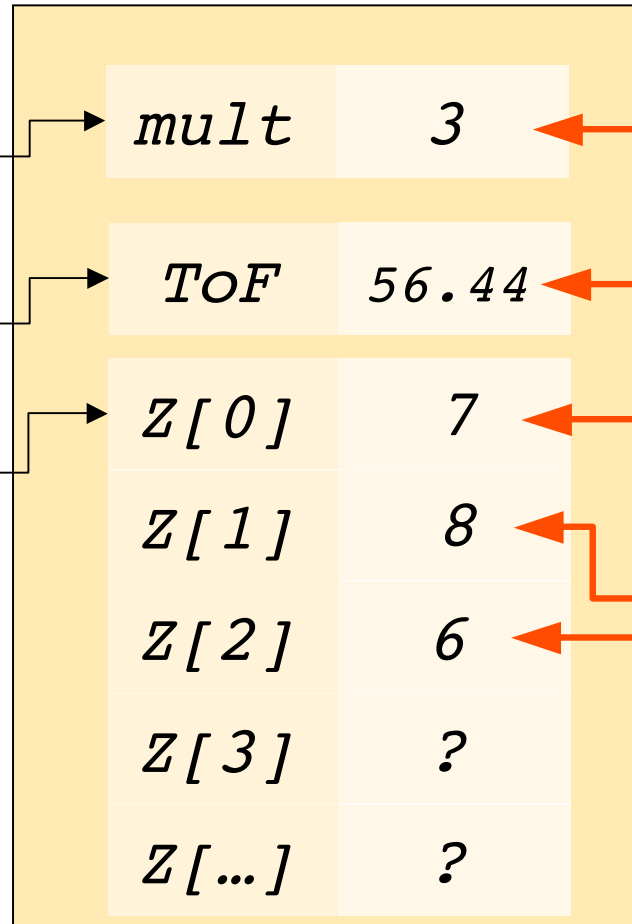
*Reading the file*

```
tree->GetEntry(5)
```

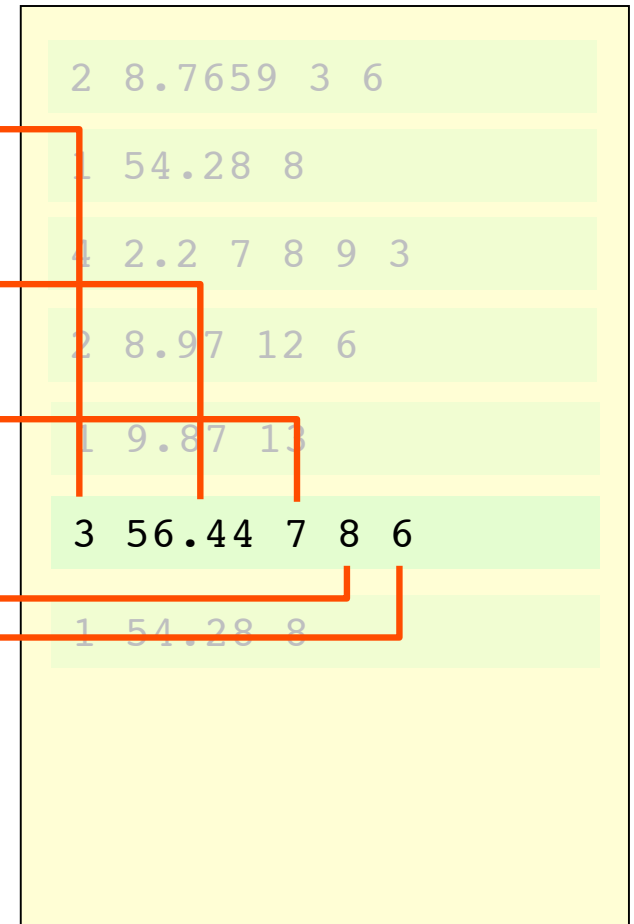
*Tree*



*Memory*



*File*



# Example: filling a tree with data

[http://caeinfo.in2p3.fr/root/Formation/en/Day4/tree\\_struc.C](http://caeinfo.in2p3.fr/root/Formation/en/Day4/tree_struc.C)  
[http://caeinfo.in2p3.fr/root/Formation/en/Day4/tree\\_struc.data](http://caeinfo.in2p3.fr/root/Formation/en/Day4/tree_struc.data)

- Have a look at the file **tree\_struc.C**
- We will use a *structure*\* : *\*it's not ROOT, it's from C !*

## Declaration

```
struct Mon_Event{
  Int_t mult;
  Float_t Z[50];
  Float_t Theta[50];
  Float_t Energie[50];
};
```

## Use

```
Mon_Event event;

event.mult = 0;
event.Z[3] = 2;
file >> event.mult;
```

*Reading data in a file*

# Example: filling a tree with data

- Declaration of the tree

```
TTree *t = new TTree("t", "TTree with a structure");
```

 *The TTree will be in the general memory (heap)*

- Declaration of a branch with an integer and three branches with variable size arrays of single precision real numbers

```
t->Branch("M_part", &event.Mult, "Mult/I");  
t->Branch("Z_part", event.Z, "Z[Mult]/F");  
t->Branch("Th_part", event.Theta, "Theta[Mult]/F");  
t->Branch("E_part", event.Energie, "Energie[Mult]/F");
```

 *The name of the branch is not necessarily the name of the variable*

 *The arrays have a variable size*

# With a single branch...

[http://caeinfo.in2p3.fr/root/Formation/en/Day4/tree\\_struc2.C](http://caeinfo.in2p3.fr/root/Formation/en/Day4/tree_struc2.C)

- Have a look at the file **tree\_struc2.C**
- Declaration of a single branch pointing to the structure

*the address of the variable*

***event** of type **Mon\_Event** is given*

*arrays have a fixed size*

```
t->Branch( "bEvent", &event,  
          "Mult/I:Z[50]/F:Theta[50]/F:Energie[50]/F" );
```

*There are many leaves (variables) on this branch*



# Example: filling a tree with data

- Data will be read in the ASCII file `tree_struc.data`

```
#include "Riostream.h"
...
ifstream file;
file.open("tree_struc.data");
...
file >> event.Mult;
...
for(Int_t i=0;i<event.Mult;i++)
{
file >> event.Z[i];
file >> event.Theta[i];
file >> event.Energie[i];
}
t->Fill();
...
file.close();
```

*Special ROOT  
declaration of input/output  
system of C++*

*opening the data file*

*Reading the data and  
filling the structure*

*the data in the structure are  
transferred to the tree*

# Looking at the tree structure

- Run the script and look at the tree !

```
.L tree_struc.C+      | TFile *f=new  
                      |     TFile("tree_struc.root")  
MakeTree()           | TTree *a=(TTree *)f->Get("t")  
                      | a->Print()  
                      |
```

```
*****  
*Tree      :t          : TTree avec une structure                               *  
*Entries   : 100000    : Total =          25750346 bytes  File Size =   16900683 *  
*          :           : Tree compression factor =    1.52                       *  
*****  
*Br       0 :M_part    : Mult/I                                              *  
*Entries   : 100000    : Total Size=      401568 bytes  File Size =     94299 *  
*Baskets   :          12 : Basket Size=      32000 bytes  Compression=    4.07 *  
*.....*  
*Br       1 :Z_part    : Z[Mult]/F                                           *  
*Entries   : 100000    : Total Size=     8449454 bytes  File Size =   1840614 *  
*Baskets   :          276 : Basket Size=      32000 bytes  Compression=    4.58 *  
*.....*  
*Br       2 :Th_part   : Theta[Mult]/F                                        *  
*Entries   : 100000    : Total Size=     8449745 bytes  File Size =   7396565 *  
*Baskets   :          276 : Basket Size=      32000 bytes  Compression=    1.14 *  
*.....*  
*Br       3 :E_part    : Energie[Mult]/F                                       *  
*Entries   : 100000    : Total Size=     8449472 bytes  File Size =   7520599 *  
*Baskets   :          276 : Basket Size=      32000 bytes  Compression=    1.12 *  
*.....*
```

# Accessing the tree data

- Looking at an "event"

a->Show(15)

```
=====> EVENT:15  
Mult          = 15  
Z             = 30,  
              34, 1, 1, 17, 1,  
              8, 2, 1, 1, 2,  
              2, 1, 1, 2  
Theta        = 14.8766,  
              10.048, 59.2787, 164.868, 8.45649, 21.6054,  
              46.5263, 28.4612, 29.1083, 72.3277, 57.2474,  
              32.4265, 16.6426, 6.97173, 9.6734  
Energie      = 983.813,  
              44.1665, 85.591, 29.5007, 655.211, 59.0234,  
              155.18, 134.403, 21.3786, 10.8284, 19.2134,  
              36.4518, 79.2352, 23.5012, 24.5475
```

*Using a tree*

# Accessing the tree data

- Selecting the events and print variables values:

```
a->Scan("Mult:Z[30]:Energie[30]","Mult>30","",1000,0)
```

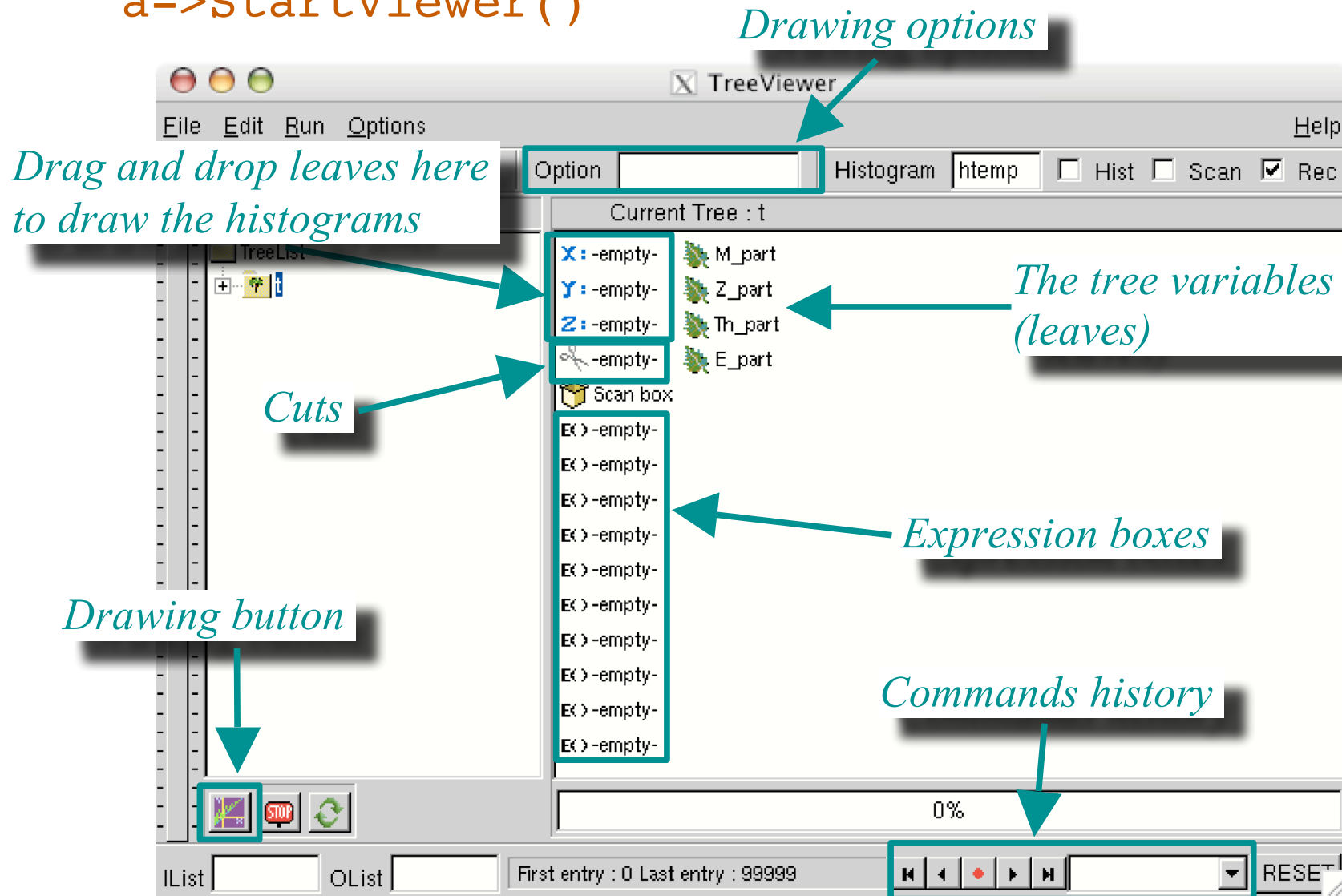
*Selection* 

*Event number* 

```
*****  
*      Row      *      Mult *      Z[30] * Energie[3 *  
*****  
*          46 *          32 *          2 * 47.778400 *  
*          95 *          31 *          2 * 48.006801 *  
*         399 *          31 *          1 * 28.520700 *  
*         461 *          31 *          2 * 67.939399 *  
*         628 *          32 *          2 * 69.046302 *  
*****  
==> 5 selected entries
```

# The graphical interface

a->StartViewer()



# For the single branch tree

(tree\_struc2.root)

The image shows a screenshot of the TreeViewer application window. The window title is "TreeViewer" and it has a menu bar with "File", "Edit", "Run", "Options", and "Help". Below the menu bar, there are several controls: a text field labeled "Option", a "Histogram" button, a text field with "htemp", and checkboxes for "Hist", "Scan", and "Rec". The main area is titled "Current Tree : t" and contains a tree structure. On the left, there is a "TreeList" panel with a vertical scrollbar and a "Drawing button" icon. The tree structure itself has a column of expression boxes on the left, some containing "-empty-", and a column of tree variables on the right, including "bEvent", "bEvent.Mult", "bEvent.Z", "bEvent.Theta", and "bEvent.Energie". At the bottom, there is a "Commands history" panel with a progress indicator showing "0%" and a "RESET" button. The interface is annotated with several labels and arrows: "Drawing options" points to the menu bar; "Drag and drop leaves here to draw the histograms" points to the TreeList; "Cuts" points to the "-empty-" boxes; "Drawing button" points to the drawing icon; "Tree branch" points to the "bEvent" node; "The tree variables (leaves)" points to the "bEvent.Energie" node; "Expression boxes" points to the "-empty-" boxes; and "Commands history" points to the bottom panel.

*Drawing options*

*Drag and drop leaves here to draw the histograms*

*Cuts*

*Drawing button*

*Tree branch*

*The tree variables (leaves)*

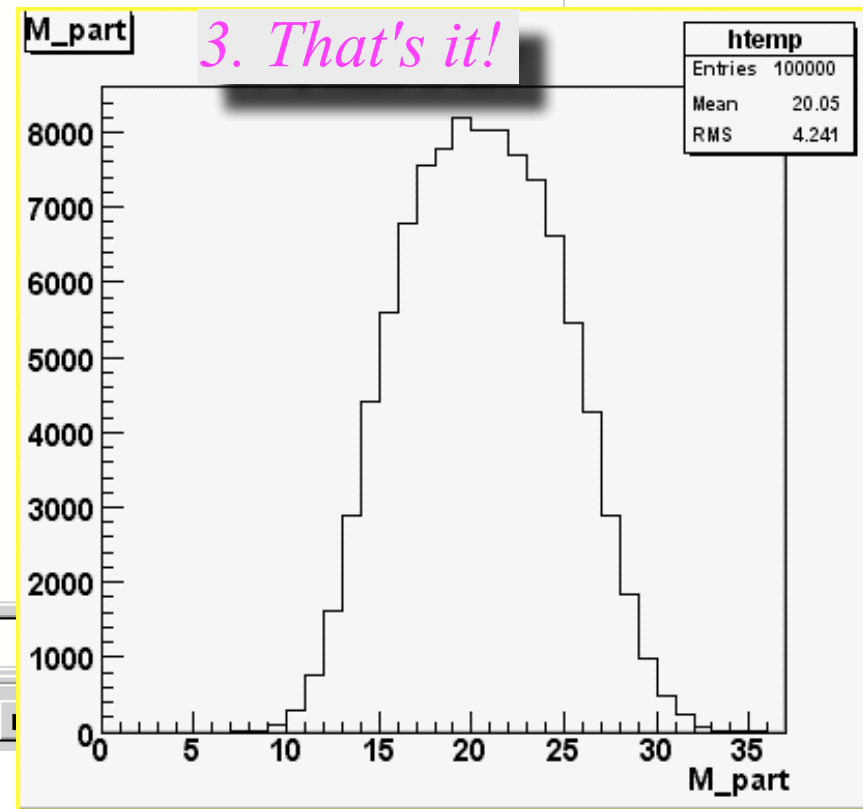
*Expression boxes*

*Commands history*

# Plotting a 1D histogram

1. Choose the variable

2. Click here





# Plotting a 2D histogram

2. Set the drawing option

The screenshot shows the TreeViewer application window. The menu bar includes File, Edit, Run, Options, and Help. The Command field is empty, and the Option field is set to 'COL'. The Histogram section has 'htemp' selected, and checkboxes for 'Hist', 'Scan', and 'Rec' are present, with 'Rec' checked. The 'Current Folder' pane on the left shows a 'TreeList' folder. The 'Current Tree' pane in the center lists several variables: '~Z\_part', '~E\_part', 'Z\_part', 'M\_part', 'Th\_part', and 'E\_part'. The 'Current Tree' pane is annotated with pink boxes and arrows: one box around '~E\_part' and 'Z\_part' with an arrow pointing to the 'Option' field, and another box around 'Z\_part' and 'E\_part' with an arrow pointing to the 'Rec' checkbox. The 'Options' menu is also indicated by a pink arrow. At the bottom left, a pink box highlights a plotting icon, with an arrow pointing to it from the text '3. Click here'. The bottom status bar shows 'IList', 'OList', and 'Content : E\_part'. On the right side, a 2D histogram plot is displayed with the title 'E\_part:Z\_part'. The y-axis is labeled 'E\_part' and ranges from 0 to 2500. The x-axis is labeled 'Z\_part' and ranges from 0 to 50. The plot shows a dense distribution of data points, colored in a gradient from red to blue. A pink callout box with the text '4. That's it! (SetLogz !!)' is overlaid on the plot.

1. Choose the variables

3. Click here

4. That's it!  
(SetLogz !!)

# Recording the current display

2. Right click here

3. Choose this item

4. Type the new name

5. Name of the record

1. Click here

The screenshot shows the 'TreeViewer' application window. The menu is open, and 'SetRecordName' is highlighted. A dialog box titled 'TTreeView::SetRecordName' is open, with the text '(const char\*) name' and a text input field containing 'E vs Z'. The playback control bar at the bottom shows a red dot on the play button and a dropdown menu with 'E vs Z' selected.

# Using cuts

3. Drag and drop the expression box in the cut box

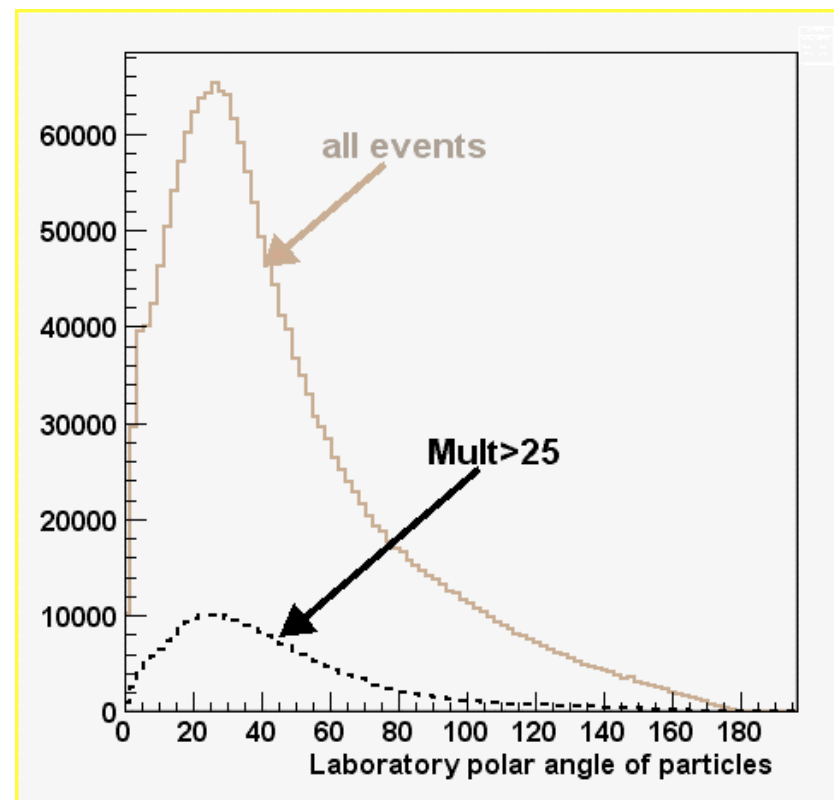
The screenshot shows a window titled "TreeView" with a menu bar (File, Edit, Run, Options, Help) and a toolbar (Histogram, htemp, Hist, Scan, Rec). Below the toolbar is a "Content Tree" showing a list of items: M\_part, Z\_part, Th\_part, E\_part, and several "-empty-" entries. A pink box highlights one of the "-empty-" entries, with a pink arrow pointing to it from the text "1. Double-click on an empty expression box E()". Below the tree is a "Scan box" and a "~HiMult" entry. A pink arrow points from the text "2. Type the cut condition and the name of its alias" to a dialog box titled "Expression editor" which has a "Selection" field containing "Mult > 25" and an "Alias" field containing "~HiMult". The dialog box also has "Cancel" and "Done" buttons. A pink arrow points from the text "3. Drag and drop the expression box in the cut box" to the "Scan box" area.

1. Double-click on an empty expression box E()

2. Type the cut condition and the name of its alias

# Using cuts (2)

- Drag and drop the **Th\_part** variable on the  $x$  axis
- Drag and drop the cut in the "scissors box"
- Double-click on the "scissors box" to disable the cut selection (red line)
- Draw the histogram **without the cut selection**
- Enable the cut selection
- Type "same" in the drawing option field
- Draw the histogram **with the cut selection**
- Record the display
- Perfect the presentation of the figure !



# Save it...

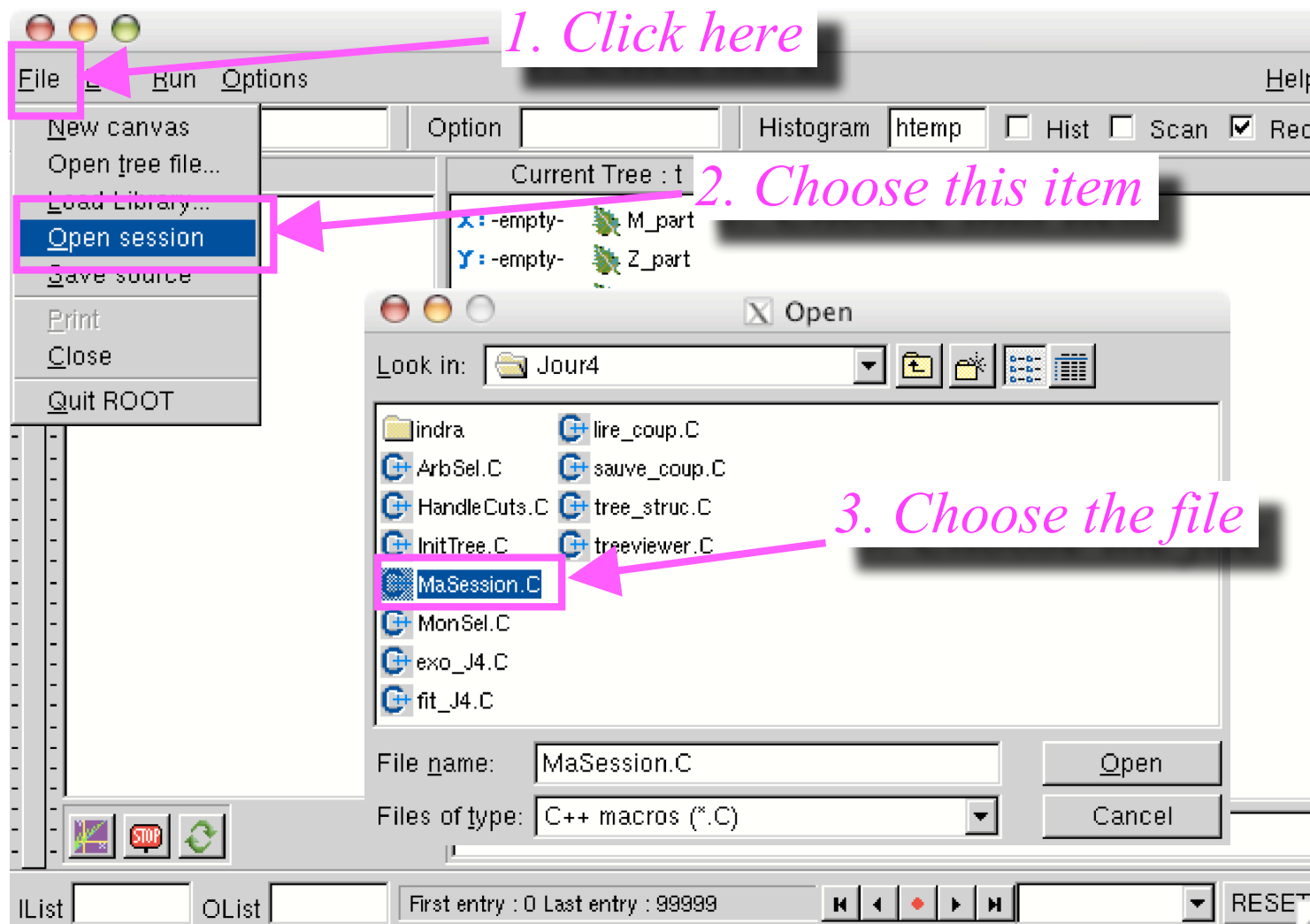
The screenshot shows a software window titled "TreeViewer" with a menu bar (File, Edit, Run, Options) and a toolbar. A context menu is open over the "Current Tree" area, listing various actions. The "SaveSource" option is highlighted in blue. A dialog box titled "TTreeView::SaveSource" is overlaid on top, with the "filename" field containing "MaSession.C" and the "option" field empty. The dialog has "OK" and "Cancel" buttons. Three pink annotations with arrows point to the context menu, the "SaveSource" option, and the filename field.

1. Right-click here

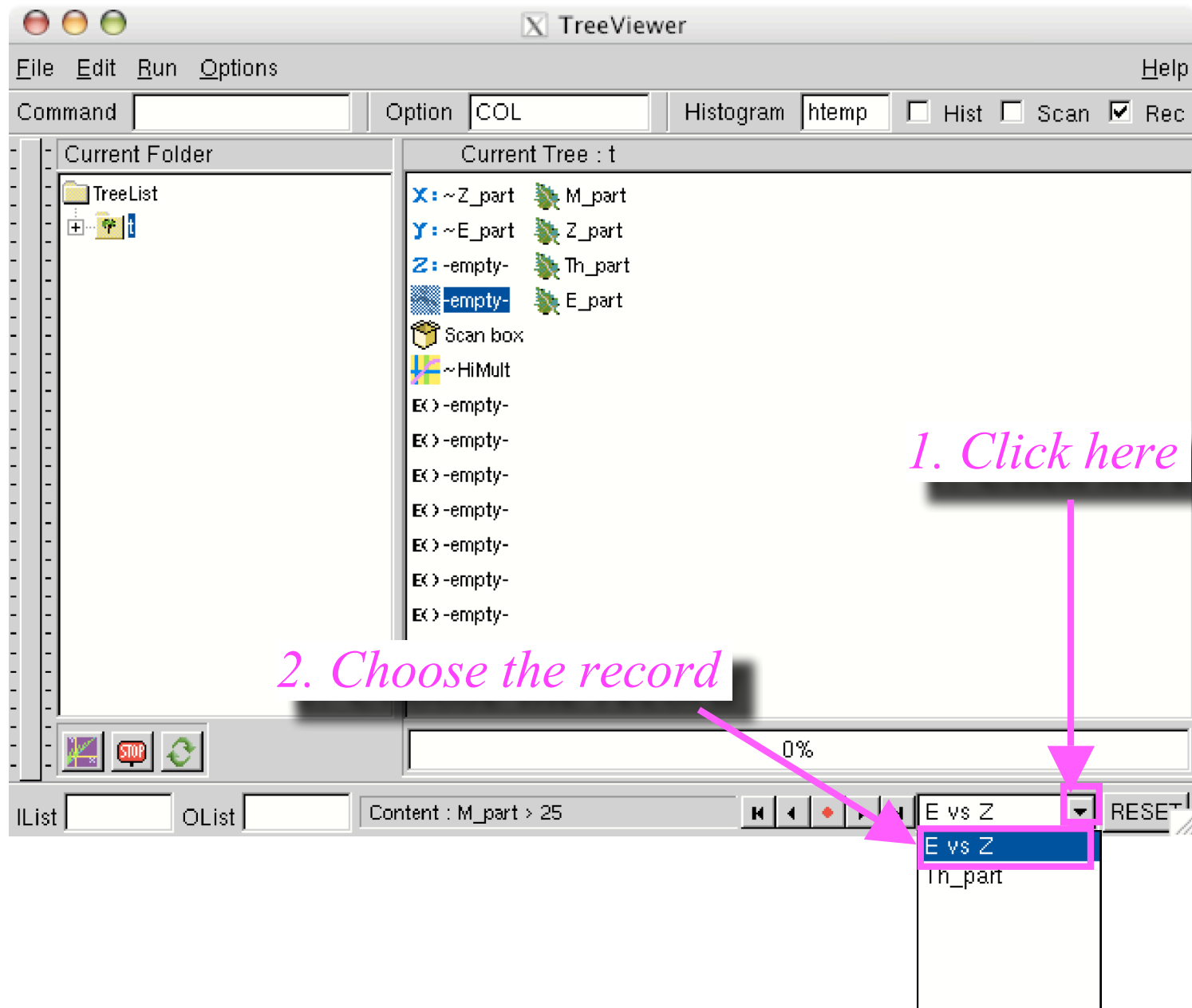
2. Choose this item

3. Type the file name

# Everything is not lost...



# Recalling a recorded display



*It's guillotine time: the cut  
machine*



# Crapphical cuts

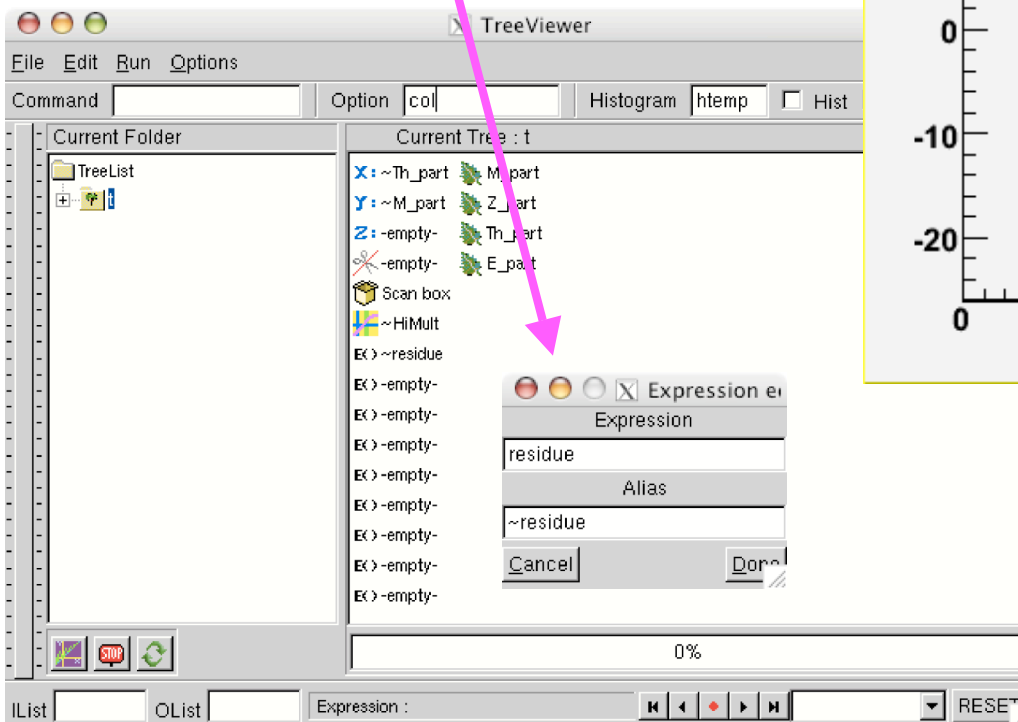
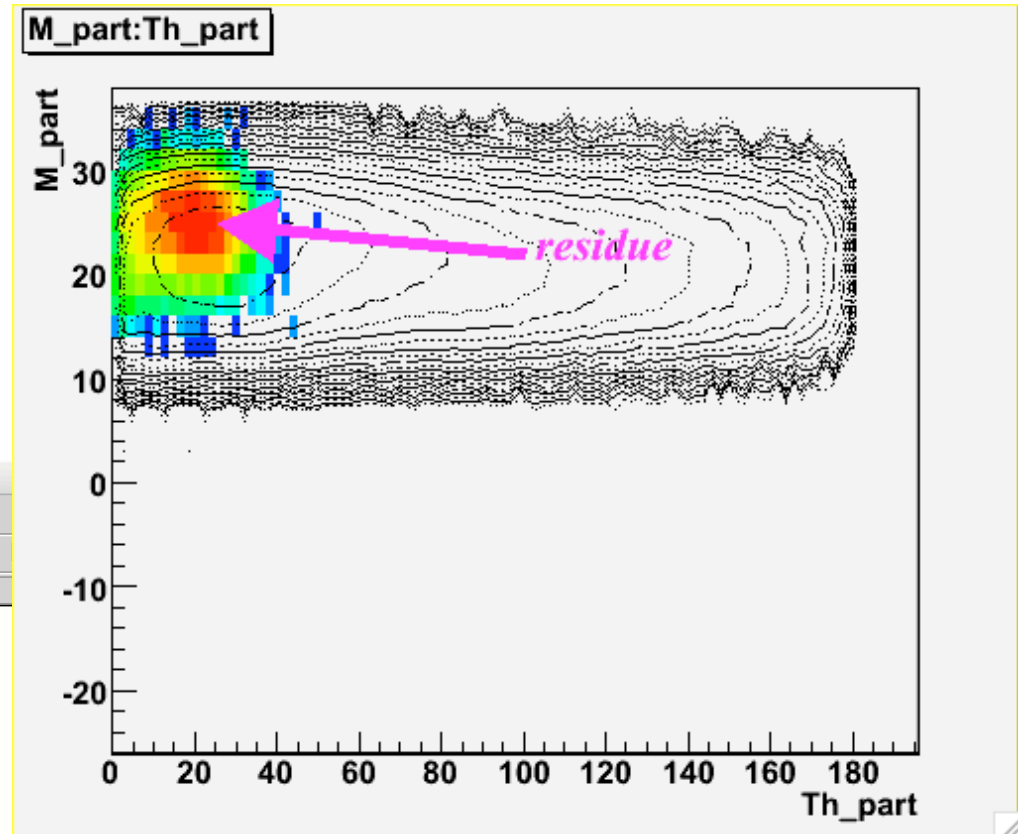
- Open the tool-bar (Canvas menu View->Toolbar)

The screenshot shows a software window titled 'c1' with a menu bar (File, Edit, View, Options, Inspect, Classes, Help) and a toolbar. A heatmap is displayed with a closed contour drawn over it. A contextual menu is open over the heatmap, listing various actions. The 'SetName' option is highlighted. A dialog box titled 'TCutG::SetName' is open, showing a text field with 'residue' and 'OK' and 'Cancel' buttons. Annotations with pink arrows and text boxes provide instructions:

1. Click here (pointing to a toolbar icon)
2. Draw a closed contour (left-click for each point, left double-click to close it) (pointing to a contour on the heatmap)
3. Right click to activate the contextual menu (pointing to a right-click on the heatmap)
4. Choose this item (pointing to 'SetName' in the menu)
5. Type the cut name (pointing to 'residue' in a dialog box)

# Using the cut

- When the name of the graphical cut is given to an expression box, this cut can be used to select events...



# Mind your fingers: let's mix our cuts

```
a->Draw("Z_part", "M_part > 30", "")
```

- But also...

```
TCut cut1("M_part > 30")
```

```
a->Draw("Z_part", cut1, "")
```

- Or...

```
TCut cut2("E_part < 200")
```

```
a->Draw("Z_part", cut1 && cut2, "")
```

*AND for C++*

- For the graphical cuts

```
a->Draw("Z_part", cut1 || "residue", "")
```

*OR for C++*

*The Swiss knife...*

# Variable combinations

- Variables can be combined to define new ones.
- Examples:

Draw the parallel velocity component  $V_z$

```
a->Draw("sqrt(E_part/(931.5*Z_part))*cos(Th_part*3.1416/180.)")
```

Draw the transverse energy as a function of Z

```
a->Draw("E_part*pow(sin(Th_part*3.1416/180.),2):Z_part", "", "box")
```

- The new variables can be defined in the expression boxes of the TreeViewer

Expression	
E_part*pow(sin(Th_part*3.1416/180),2)	
Alias	
~Etrans	
<input type="button" value="Cancel"/>	<input type="button" value="Done"/>

# Alias, poor Yorick...

- Pseudo variables (alias) can be defined

## Examples:

velocity modulus:

```
a->SetAlias("V", "sqrt(E_part/(931.5*Z_part))*30")
```

cosine of the  $\theta$  angle:

```
a->SetAlias("cost", "cos(Th_part*3.1416/180.)")
```

$V_z$  velocity component

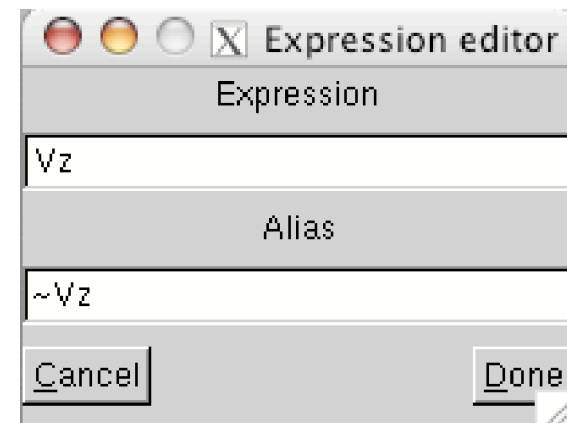
```
a->SetAlias("Vz", "V*cost")
```

## Use:

```
a->Draw("Z_part:Vz", "Vz>-10", "col")
```

- They can be used in the TreeViewer

**BEWARE:** an alias from the TreeViewer can not be used with the draw command **a->Draw()**



# Summing everything...

- Macro-commands can be used with arrays in trees:

## Examples:

Sum of products  $Z*Vz$ :

```
a->Draw("Sum$(Z*Vz)")
```

Alias Mimf

```
a->SetAlias("Mimf","Sum$(Z>2)")
```

Z	6	1	4	2	Sum\$(Z>2)
Z>2	1	0	1	0	2

Alias Transverse Energy of light particles

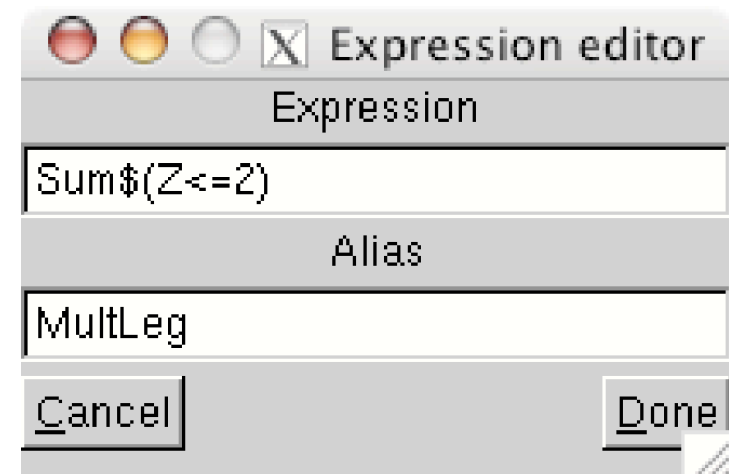
```
a->SetAlias("Et12","Sum$(E*(1-cost*cost)*(Z<=2))")
```

## Use:

```
! a->Draw("Mimf:Et12","Sum$(Z>2)>3","col")  
a->Draw("Mimf:Et12","Mimf>3","col")
```

- These macro-commands can be used in the TreeViewer
- Have a look at other macro-commands at

<http://root.cern.ch/root/html/TTree.html#TTree:Draw>



# Strings

- Character strings can be passed as arguments of Draw, Scan, SetAlias, GetAlias.

## Examples:

We want to define alias names "NewVarX" as follows:

"variableX-(maximum of the histogram named HistoX\_mono)"  
for X ranging from 1 to 10

```
Char_t nomAlias[80];
for(Int_t i=1;i<=10;i++)
{
    sprintf(nomAlias, "NewVar%d", i);
    TString var("variable");
    var+=i;
    TH1 *h=(TH1 *)gROOT->FindObject(Form("Histo%d_mono", i));
    Double_t y=h->GetMaximum();
    a->SetAlias(nomAlias, Form("%s-%f", var.Data(), y));
}
a->GetListOfAliases()->ls();
```



# Projection to a histogram

Creation of the histogram:

```
TH1F *h1=new TH1F("DistZ",  
                  "Distribution de charge",100,-0.5,99.5)
```

- Projection!

*the name of the histogram is necessary!*

```
a->Draw("Z_part >> DistZ", "M_part>30")
```

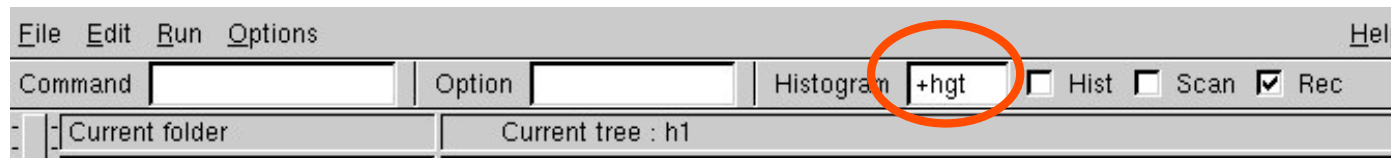
ou

```
a->Project("DistZ", "Z_part", "M_part>30")
```

- Cumulative projection !

```
a->Draw("Z_part >>+DistZ", "M_part<=30")
```

or a "+" sign before the histogram name in the TreeViewer



# The event lists

- These lists can save time if a complex and time consuming cut is applied frequently: only the index numbers of the events corresponding to this cut are recorded in the list!

```
a->Draw(">> listem", "M_part>30", "")
```

- To use the event list:

```
TEventList *lm=(TEventList *)gROOT->FindObject("listem")
```

```
lm->Print("all") ← Printout of all event numbers in the list
```

```
a->SetEventList(lm)
```

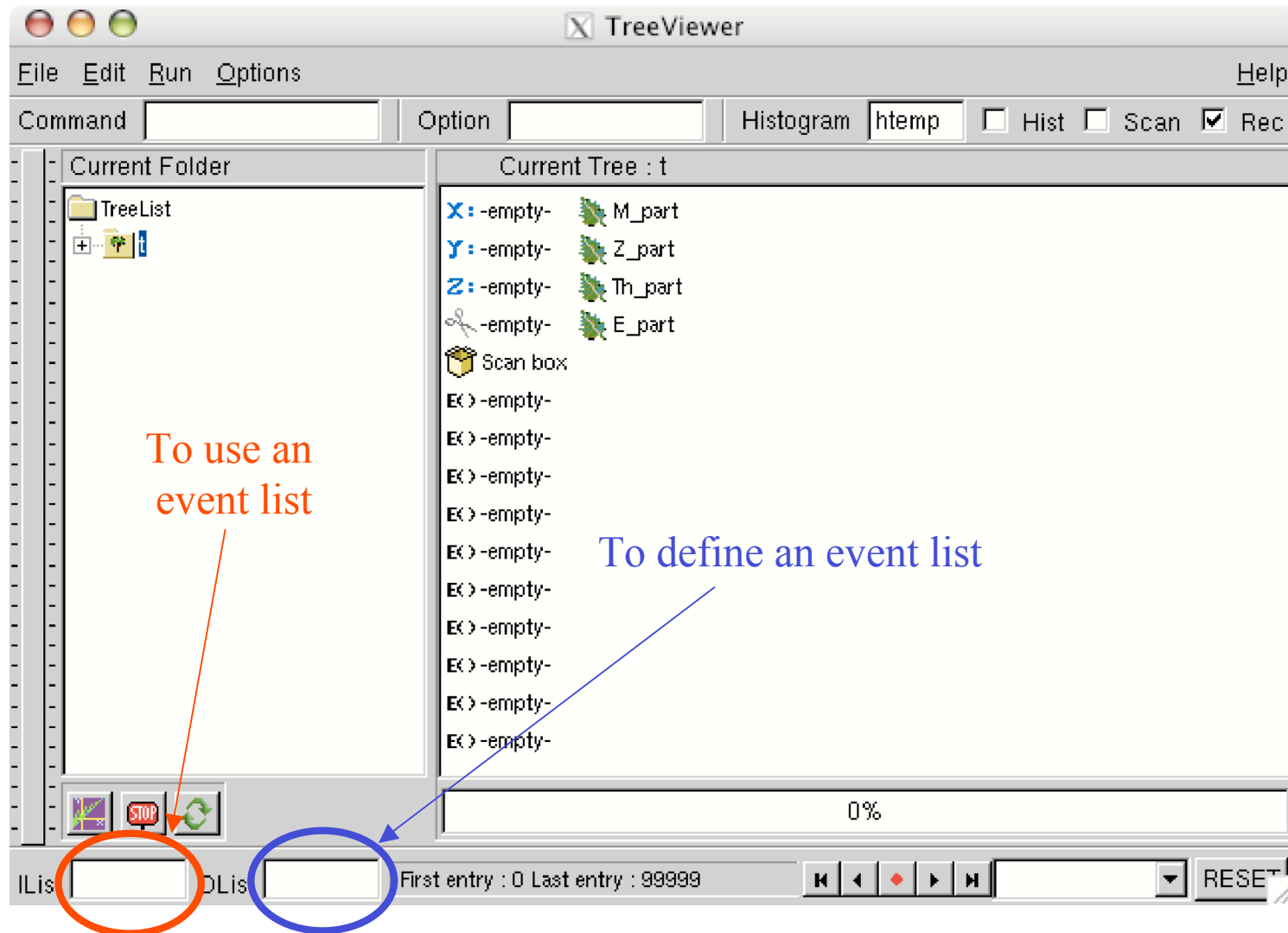
```
a->Draw("Z_part")
```

```
a->Draw("E_part")
```

- To remove it from the tree:

```
a->SetEventList(0)
```

# The event list in the TreeViewer



# Exercise

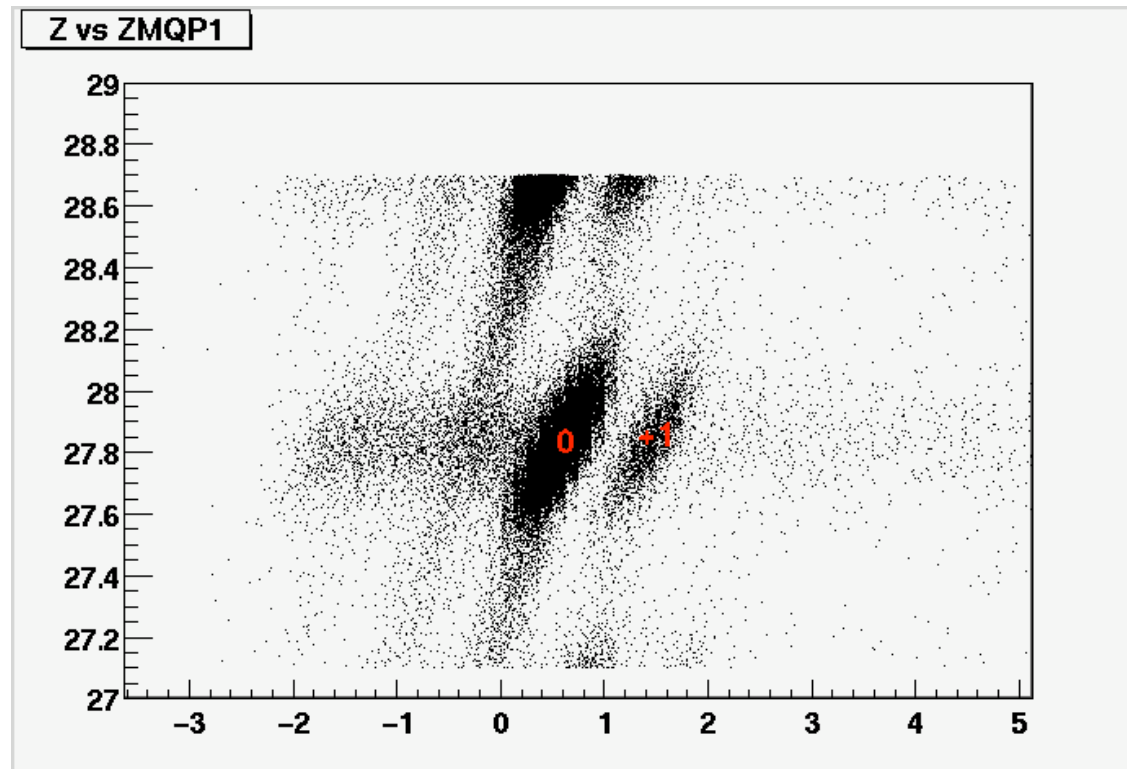
- You will analyse data from a LISE\* experiment whose goal is to show the differences between the  $\gamma$  energy spectra for two nickel isotopes. The data are stored in a TTree in the file **r50\_69ni.root**.
- You will proceed step by step:
  1. Selection of the correct charge state
  2. Selection of the two Ni isotopes.
  3. Calibrate time spectra to build a cumulative histogram.
  4. Building the  $\gamma$  energy spectra for both isotopes.

[http://caeinfo.in2p3.fr/root/Formation/en/Day4/r50\\_69ni.root](http://caeinfo.in2p3.fr/root/Formation/en/Day4/r50_69ni.root)

*\*Thanks to M.Sawicka, F.De Oliveira and J.M.Daugas!*

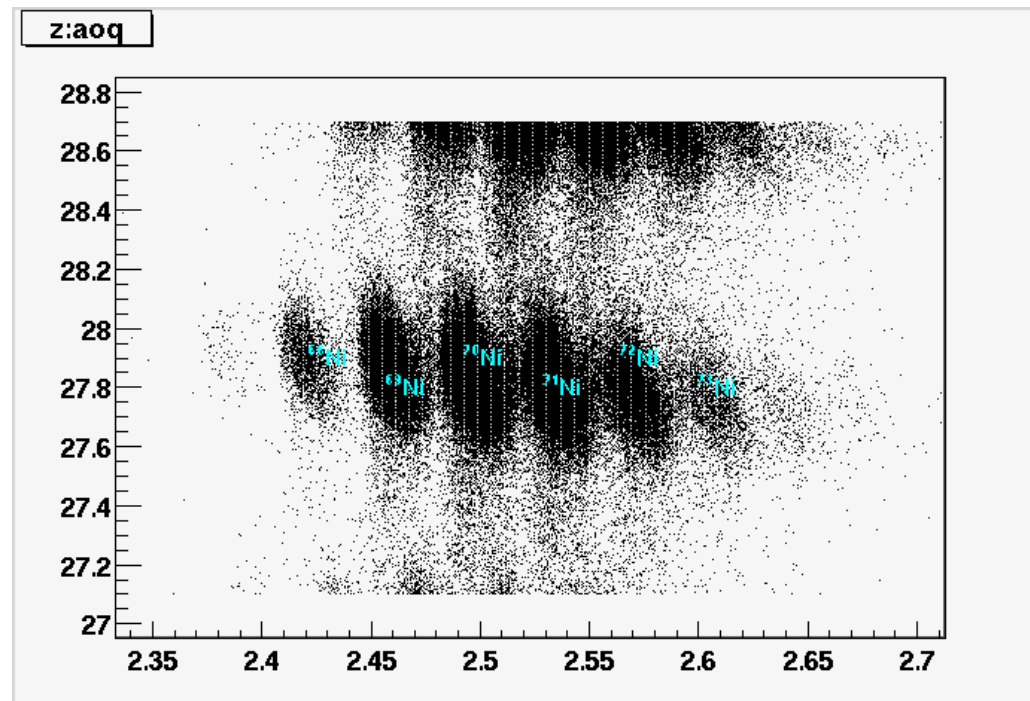
# Exercise: Step 1

- Selection of the charge state
  - Build the histogram **z** versus **zmqp1**.
  - Build a graphical cut named CUTEc around the accumulation of data centred at (0.5,27.8)



# Exercise: Step 2

- Selection of the Ni isotopes:
  - Build the histogram **z** versus **aoq**
  - Build a graphical cut named CUTNI69 around the area centred at (2.45,27.9)
  - Build a graphical cut named CUTNI70 around the area centred at (2.5,27.9)



# *Saving the cuts*

```
#include "TFile.h"
#include "TCUTG.h"

void SaveCuts(void)
{
  TFile *fcoup=new TFile("coupures.root","recreate");
  fcoup->cd();
  gROOT->FindObject("CUTEC")->Write();
  gROOT->FindObject("CUTNI69")->Write();
  gROOT->FindObject("CUTNI70")->Write();
  fcoup->Close();
}
```

<http://caeinfo.in2p3.fr/root/Formation/en/Day4/HandleCut.C>

# *To retrieve the cuts*

```
void LoadCuts(void)
{
TFile *fcoup=new TFile("coupures.root");
TCutG *CUTEK=(TCUTG *)fcoup->Get("CUTEK");
TCutG *CUTNI69=(TCUTG *)fcoup->Get("CUTNI69");
TCutG *CUTNI70=(TCUTG *)fcoup->Get("CUTNI70");
fcoup->Close();
}
```

*Use:*

```
root[0] .L HandleCuts.C+
```

```
root[1] SaveCuts() to save them
```

```
root[2] LoadCuts() to load them
```



## *Exercise: Step 3*

- Calibrating the « long » time spectra.
  - Build the histogram of **tg1lo** for values of **tg1lo** lower than 3000
  - Locate the abscissa **T1M** of the spectrum's maximum
  - Build the alias named **RTG1LO** = **tg1lo** - **T1M**
  - Repeat the same procedure for the 5 other variables **tgxlo** for **x** ranging from 2 to 6.

# Exercise: Step 4

- **Build the following histograms for the charge state 0**
  - Cumulative histogram of spectra **Egxc** for **x** ranging from 1 to 6
  - Same histogram for  $^{69}\text{Ni}$  alone
  - Same histogram for  $^{70}\text{Ni}$  alone
  - Superimpose these histograms
  - Conclusions?
- **Build the following histograms for the charge state 0**
  - Cumulative histogram of spectra **Egxc** vs RTGXLO for **x** ranging from 1 to 6 for the  $^{70}\text{Ni}$ .
  - Make the projections of this histogram on the time axis for the two most intense energy peaks ( $E_\gamma \approx 183 \text{ keV}$  et  $E_\gamma \approx 447 \text{ keV}$ )
  - Extract from these projections the half-life of these two  $\gamma$  peaks (using a fit function being the sum of a constant and an exponential)
  - Conclusions?