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*...

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

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

- **tree name**
- **tree title**
- **tree branches contain the variables (leaves)**
- **Name and type of the variable**
- **variable address in the memory**
- **name of the branch**
Defining the branches

• Simple variables

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

• Fixed size array

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

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

• Variable size array

```cpp
tree->Branch("Mult", &mult, "mult/I");
tree->Branch("dM/dZ", Z, "Z[mult]/D");
```
What happens in memory...

Tree

- anInteger Mult
- aDouble TdV
- dM/dZ Z[mult]

Memory

- mult
-ToF
- Z[0]
- Z[1]
- Z[2]
- Z[3]
- Z[...]

Z[mult]
What happens in memory...

Writing to the file

Tree

- anInteger Mult
- aDouble TdV
- dM/dZ Z[mult]

Memory

- mult=2
- ToF=8.7659
- Z[0]=3
- Z[1]=6

File

- mult 2
- ToF 8.7659
- Z[0] 3
- Z[1] 6
- Z[2] ?
- Z[3] ?
- Z[...] ?
What happens in memory...

Writing to the file

Tree

- anInteger Mult
- aDouble TdV
- dM/dZ Z[mult]

Memory

- mult 2
- ToF 8.7659
- Z[0] 3
- Z[1] 6
- Z[2] ?
- Z[3] ?
- Z[...] ?

tree->Fill()

File

2 8.7659 3 6
What happens in memory...

Writing to the file

Tree

- anInteger Mult
- aDouble TdV
- dM/dZ Z[mult]

Memory

<table>
<thead>
<tr>
<th>mult</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>ToF</td>
<td>54.28</td>
</tr>
<tr>
<td>Z[0]</td>
<td>8</td>
</tr>
<tr>
<td>Z[1]</td>
<td>6</td>
</tr>
<tr>
<td>Z[2]</td>
<td>?</td>
</tr>
<tr>
<td>Z[3]</td>
<td>?</td>
</tr>
<tr>
<td>Z[...]</td>
<td>?</td>
</tr>
</tbody>
</table>

File

2 8.7659 3 6

Tree diagram with nodes labeled as follows:
- anInteger
- aDouble
- dM/dZ

Memory section with variables:
- mult
- ToF
- Z[0]
- Z[1]
- Z[2]
- Z[3]
- Z[...]

File section containing:
2 8.7659 3 6
What happens in memory...

Writing to the file

tree->Fill()

Tree

- anInteger Mult
- aDouble TdV
- dM/dZ Z[mult]

Memory

- mult 1
- ToF 54.28
- Z[0] 8
- Z[1] 6
- Z[2] ?
- Z[3] ?
- Z[...] ?

File

2 8.7659 3 6
1 54.28 8
What happens in memory...

Tree

- anInteger: Mult
- aDouble: TdV
- dM/dZ: Z[mult]

Memory

- mult: 2
- ToF: 8.7659
- Z[0]: 3
- Z[1]: 6
- Z[2]: ?
- Z[3]: ?
- Z[...]: ?

File

- 2 8.7659 3 6
- 1 54.28 8
- 4 2.2 7 8 9 3
- 2 8.97 12 6
- 1 9.87 13
- 3 36.44 7 8 6
- 1 54.28 8

Reading the file

tree->GetEntry(0)
What happens in memory...

Reading the file

Tree
- anInteger Mult
- aDouble TdV
- dM/dZ Z[mult]

Memory
- mult 1
- ToF 54.28
- Z[0] 8
- Z[1] 6
- Z[2] ?
- Z[3] ?
- Z[...] ?

File

```
2  8.7659  3  6
1  54.28  8
4  2.3   7  8  9  3
2  8.97  12  6
1  9.87  13
3  56.44  7  8  6
1  54.28  8
```
What happens in memory...

Reading the file

Tree

- anInteger  Mult
- aDouble  TdV
- dM/dZ  Z[mult]

Memory

- mult  3
- ToF  56.44
- Z[0]  7
- Z[1]  8
- Z[3]  ?
- Z[...]  ?

File

2 8.7659 3 6
54.28 8
2.2 7 8 9 3
2 8.97 12 6
1 9.87 13
3 56.44 7 8 6
1 54.28 8
Example: filling a tree with data


- Have a look at the file tree_struc.C
- We will use a structure*

*it’s not ROOT, it’s from C!

Declaration

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

Use

```c
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

```cpp
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

```cpp
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...


• Have a look at the file `tree_struc2.C`

• Declaration of a single branch pointing to the structure:

```c
// 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`

```cpp
#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!

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

<table>
<thead>
<tr>
<th>*Tree</th>
<th>:t</th>
<th>TTree avec une structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>*Entries</td>
<td>100000</td>
<td>Total = 25750346 bytes File Size = 16900683</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tree compression factor = 1.52</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>*Br</th>
<th>0 :M_part</th>
<th>Multi/I</th>
</tr>
</thead>
<tbody>
<tr>
<td>*Entries</td>
<td>100000</td>
<td>Total Size= 401568 bytes File Size = 94299</td>
</tr>
<tr>
<td>*Baskets</td>
<td>12</td>
<td>Basket Size= 32000 bytes Compression= 4.07</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>*Br</th>
<th>1 :Z_part</th>
<th>Z[Multi]/F</th>
</tr>
</thead>
<tbody>
<tr>
<td>*Entries</td>
<td>100000</td>
<td>Total Size= 8449454 bytes File Size = 1840614</td>
</tr>
<tr>
<td>*Baskets</td>
<td>276</td>
<td>Basket Size= 32000 bytes Compression= 4.58</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>*Br</th>
<th>2 :Th_part</th>
<th>Theta[Multi]/F</th>
</tr>
</thead>
<tbody>
<tr>
<td>*Entries</td>
<td>100000</td>
<td>Total Size= 8449745 bytes File Size = 7396565</td>
</tr>
<tr>
<td>*Baskets</td>
<td>276</td>
<td>Basket Size= 32000 bytes Compression= 1.14</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>*Br</th>
<th>3 :E_part</th>
<th>Energie[Multi]/F</th>
</tr>
</thead>
<tbody>
<tr>
<td>*Entries</td>
<td>100000</td>
<td>Total Size= 8449472 bytes File Size = 7520599</td>
</tr>
<tr>
<td>*Baskets</td>
<td>276</td>
<td>Basket Size= 32000 bytes Compression= 1.12</td>
</tr>
</tbody>
</table>
Accessing the tree data

• Looking at an "event"

\[ \text{a}\rightarrow\text{Show}(15) \]

\[
\begin{align*}
\text{EVENT:15} \\
\text{Mult} \quad & = 15 \\
\text{Z} \quad & = 30, \\
& \quad 34, 1, 1, 17, 1, \\
& \quad 8, 2, 1, 1, 2, \\
& \quad 2, 1, 1, 2 \\
\text{Theta} \quad & = 14.8766, \\
& \quad 10.048, 59.2787, 164.868, 8.45649, 21.6054, \\
& \quad 46.5263, 28.4612, 29.1083, 72.3277, 57.2474, \\
& \quad 32.4265, 16.6426, 6.97173, 9.6734 \\
\text{Energie} \quad & = 983.813, \\
& \quad 44.1665, 85.591, 29.5007, 655.211, 59.0234, \\
& \quad 155.18, 134.403, 21.3786, 10.8284, 19.2134, \\
& \quad 36.4518, 79.2352, 23.5012, 24.5475
\end{align*}
\]
Using a tree
Accessing the tree data

- Selecting the events and print variables values:

\[
a \rightarrow \text{Scan(} "\text{Mult:Z[30]:Energie[30]}, "\text{Mult>30}, "", 1000, 0)\text{)}
\]

---

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>* 46 *</td>
<td>32  *</td>
<td>2 * 47.778400 *</td>
<td></td>
<td></td>
</tr>
<tr>
<td>* 95 *</td>
<td>31  *</td>
<td>2 * 48.006801 *</td>
<td></td>
<td></td>
</tr>
<tr>
<td>* 399 *</td>
<td>31  *</td>
<td>1 * 28.520700 *</td>
<td></td>
<td></td>
</tr>
<tr>
<td>* 461 *</td>
<td>31  *</td>
<td>2 * 67.939399 *</td>
<td></td>
<td></td>
</tr>
<tr>
<td>* 628 *</td>
<td>32  *</td>
<td>2 * 69.046302 *</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

=> 5 selected entries
The graphical interface

a->StartViewer()

Drag and drop leaves here to draw the histograms

The tree variables (leaves)

Drawing options

Expression boxes

Cuts

Drawing button

Commands history
For the single branch tree (tree_struc2.root)

- Drag and drop leaves here to draw the histograms
- Drawing options
- Tree branch
- The tree variables (leaves)
- Expression boxes
- Cuts
- Drawing button
- Commands history
Plotting a 1D histogram

1. Choose the variable
2. Click here
3. That's it!
Plotting a 2D histogram

1. Choose the variables

2. Set the drawing option

3. Click here

4. That's it! (SetLogz !!)
Recording the current display

1. Click here
2. Right click here
3. Choose this item
4. Type the new name
5. Name of the record
Using cuts

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

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

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

1. Right-click here
2. Choose this item
3. Type the file name
Everything is not lost...

1. Click here

2. Choose this item

3. Choose the file
Recalling a recorded display

1. Click here

2. Choose the record
It’s guillotine time: the cut machine
Graphical cuts

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

1. Click here
2. Draw a closed contour (left-click for each point, left double-click to close it)
3. Right click to activate the contextual menu
4. Choose this item
5. Type the cut name
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, ",")

• For the graphical cuts
  a->Draw("Z_part", cut1 || "residue", ",")

**AND for C++**

**OR for C++**
The Swiss knife...
Variable combinations

• Variables can be combined to define new ones.

• Examples:
  
  Draw the parallel velocity component \( V_z \)
  
  \[ a \rightarrow \text{Draw}(\sqrt{E_{\text{part}}/(931.5*Z_{\text{part}}))}*\cos(\text{Th}_{\text{part}}*3.1416/180.)) \]

  Draw the transverse energy as a function of Z
  
  \[ a \rightarrow \text{Draw}(E_{\text{part}}*\text{pow}(sin(\text{Th}_{\text{part}}*3.1416/180.),2):Z_{\text{part}},"","box") \]

• The new variables can be defined in the expression boxes of the TreeViewer
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 θ 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 $Z*Vz$:

  Sum of products $Z*Vz$:
  
  $$a->\text{Draw}("\text{Sum$(Z*Vz)$}\")$$

  Alias Mirmf
  
  $$a->\text{SetAlias}("\text{Mirmf","Sum$(Z>2)"}\")$$

  Alias Transverse Energy of light particles
  
  $$a->\text{SetAlias}("\text{Et12","Sum$(E*(1-cost*cost)*(Z<=2))"}\")$$

  Use:
  
  !a->\text{Draw}("\text{Mirmf:Et12","Sum$(Z>2)>3","col"}\")

  a->\text{Draw}("\text{Mirmf:Et12","Mirmf>3","col"}\")

- These macro-commands can be used in the TreeViewer

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

```c
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));
}
```

```c
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!**

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

    or

    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")
  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

To use an event list

To define an event list
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.


*Thanks to M.Sawicka, F.De Oliveira and J.M.Daugas!
Exercise: Step 1

• Selection of the charge state
  – Build the histogram $z$ versus $zmqpl$.
  – 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)
#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();
}

To retrieve the cuts

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

Use:

```c
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 $tg_{1lo}$ for values of $tg_{1lo}$ lower than 3000
  - Locate the abscissa $T1M$ of the spectrum’s maximum
  - Build the alias named $RTG1LO = tg_{1lo} - T1M$
  - Repeat the same procedure for the 5 other variables $tg_{xlo}$ for $x$ ranging from 2 to 6.
Exercise: Step 4

• **Build the following histograms for the charge state 0**
  – Cumulative histogram of spectra $E_{\gamma \times c}$ for $x$ ranging from 1 to 6
  – Same histogram for $^{69}$Ni alone
  – Same histogram for $^{70}$Ni alone
  – Superimpose these histograms
  – Conclusions?

• **Build the following histograms for the charge state 0**
  – Cumulative histogram of spectra $E_{\gamma \times c}$ vs RTGxLO for $x$ ranging from 1 to 6 for the $^{70}$Ni.
  – Make the projections of this histogram on the time axis for the two most intense energy peaks ($E_\gamma \approx 183$ keV et $E_\gamma \approx 447$ 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?