Introduction to Root program: L3

Presented by

DR. MOHAMMED ATTIA MAHMOUD

- PhD, Fayoum University, Egypt and Antwerp University, Belgium.
- Researcher in ENHEP, ASRT, Fayoum Uni, and BUE.
- FSQ Gen-Contact, CMS experiment, CERN, Geneva, Switzerland.
### What do we learn from this Macro

<table>
<thead>
<tr>
<th>How to</th>
<th>Commands</th>
</tr>
</thead>
</table>
| Create a Root file?             | TFile *f = new TFile("basic.root","RECREATE");  
//option: NEW, CREATE, RECREATE, UPDATE, or READ  
//Book and fill histograms and trees  
//------  
f->Write(); //write the file  
f->Close(); //close the file |
| Book and fill a histogram?      | TH1F *h1 = new TH1F("h1","x distribution",100,-4,4);  
/*do some calculation and get the parameter that you want to fill*/  
h1->Fill(x); |
| Book and fill a tree?           | TNtupled *ntuple = new TNtuple("ntuple","data from ascii file","x:y:z");  
/*do some calculation and get the parameter that you want to fill*/  
ntuple->Fill(x,y,z); |
| CINT Data types                 | Int_t and Float_t  
(see http://root.cern.ch/root/html/ListOfTypes.html ) |
"SAME": Superimpose on previous picture in the same pad.
"CYL": Use cylindrical coordinates.
"POL": Use polar coordinates.
"SPH": Use spherical coordinates.
"PSR": Use pseudo-rapidity/phi coordinates.
"LEGO": Draw a lego plot with hidden line removal.
"LEGO1": Draw a lego plot with hidden surface removal.
"LEGO2": Draw a lego plot using colors to show the cell contents.
"SURF": Draw a surface plot with hidden line removal.
"SURF1": Draw a surface plot with hidden surface removal.
"SURF2": Draw a surface plot using colors to show the cell contents.
"SURF3": Same as SURF with a contour view on the top.
"SURF4": Draw a surface plot using Gouraud shading.
“SURF5”: Same as SURF3 but only the colored contour is drawn.
**Canvas: An area mapped to a window**

<table>
<thead>
<tr>
<th>Command</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>c1 = new TCcanvas(&quot;c1&quot;,&quot;Title, w, h&quot;)</code></td>
<td>Creates a new canvas with width equal to <code>w</code> number of pixels and height equal to <code>h</code> number of pixels.</td>
</tr>
<tr>
<td><code>c1-&gt;Divide(2,2);</code></td>
<td>Divides the canvas to 4 pads.</td>
</tr>
<tr>
<td><code>c1-&gt;cd(3)</code></td>
<td>Select the 3rd Pad</td>
</tr>
<tr>
<td><code>c1-&gt;SetGridx();</code></td>
<td>You can set grid along x and y axis.</td>
</tr>
<tr>
<td><code>c1-&gt;SetGridy();</code></td>
<td></td>
</tr>
<tr>
<td><code>c1-&gt;SetLogy();</code></td>
<td>You can also set log scale plots.</td>
</tr>
</tbody>
</table>
root [1] c1 = new TCanvas("c1","Title",800,600);
root [2] c1->Divide(2,2);
root [3] c1->cd(1);
root [4] h1->Draw();
root [5] c1->cd(2);
root [6] h2->Draw();
root [7] c1->cd(3);
root [8] h1->SetLineColor(2)
root [9] h2->SetLineColor(4)
root [10] h1->Draw();
root [12] c1->cd(4);
root [13] h1->Fit("gaus");
Which fitting functions?

- The predefined functions:
  - "gaus" = $p_0 \exp(-0.5 \times \text{pow}((x-p_1)/p_2,2))$
  - "expo" = $\exp(p_0+p_1 \times x)$
  - "polN" = $p_0 + p_1 \times x + p_2 \times \text{pow}(x,2) + p_3 \times ...$
  - "landau" (guess the formula!)

How to obtain the values of the fit parameters?

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

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

Only the function name is known!
Including Parameters

```cpp
TF1 *ft = new TF1("f3", ", [0]*sqrt([1]*(sin([2]*x)+2)) /([3]+pow(x,2))", 0, 10)
ft->SetParameters(1,1,3,5)
ft->Draw()
```

```
<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>content</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>5</td>
</tr>
</tbody>
</table>
```

```cpp
TH1F *hd = new TH1F("h2","Un exemple", 100, 0, 10)
hd->FillRandom("f3",100000)
ft->SetParameters
(h2->GetMaximum(),1,2.8,6.)
hd->Fit("f3")
```

```
<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>content</td>
<td>h2-&gt;GetMaximum()</td>
<td>1</td>
<td>2.8</td>
<td>6</td>
</tr>
</tbody>
</table>
```
ROOT Trees

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

If we do not use TTree, we need to:

✓ read each event in its entirety into memory
✓ extract the parameters from the event
✓ Compute quantities from the same
✓ fill a histogram
Create Trees/TNtuple

Tfile *F = new Tfile("test.root",RECREATE);
TTree *T = new TTree("T","test");
T->Branch("x", &x,"x/F");
T->Branch("y", &y,"x/F");
T->Branch("z", &z,"x/F");
// Read/or calculate x,y and z
T->Fill();
T->Close();
F->Close();

Tfile *F = new Tfile("test.root",RECREATE);
TNtuple *T = new TNtuple("ntuple","data from ascii file","x:y:z");
// Read/or calculate x,y and z
T->Fill(x,y,z);
T->Close();
F->Close();

Draw: T->Draw("x");

T -> Print(); //print contents of root file
A `TTree` can contain integers, real numbers, **structures**, even **objects**...

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

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

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

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

• Fixed size array

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

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

• Variable size array

    tree->Branch("Mult", &mult, "mult/I");
    tree->Branch("dM/dZ", Z, "Z[mult]/D");
## Useful command in using tree

<table>
<thead>
<tr>
<th>Command</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>T-&gt;Print();</td>
<td>Prints the content of the tree</td>
</tr>
<tr>
<td>T-&gt;Scan();</td>
<td>Scans the rows and columns</td>
</tr>
<tr>
<td>T-&gt;Draw(&quot;x&quot;);</td>
<td>Draw a branch of tree</td>
</tr>
<tr>
<td>How to apply cuts:</td>
<td></td>
</tr>
<tr>
<td>T-&gt;Draw(&quot;x&quot;,&quot;x&gt;0&quot;);</td>
<td>Draw “x” when “x&gt;0”</td>
</tr>
<tr>
<td>T-&gt;Draw(&quot;x&quot;,&quot;x&gt;0 &amp;&amp; y&gt;0&quot;);</td>
<td>Draw “x” when both x &gt;0 and y &gt;0</td>
</tr>
<tr>
<td>T-&gt;Draw(&quot;y&quot;,&quot;&quot;,&quot;same&quot;);</td>
<td>Superimpose “y” on “x”</td>
</tr>
<tr>
<td>T-&gt;Draw(&quot;y:x&quot;);</td>
<td>Make “y vs x” 2d scatter plot</td>
</tr>
<tr>
<td>T-&gt;Draw(&quot;z:y:x&quot;);</td>
<td>Make “z:y:x” 3d plot</td>
</tr>
<tr>
<td>T-&gt;Draw(&quot;sqrt(x<em>x+y</em>y)&quot; );</td>
<td>Plot calculated quantity</td>
</tr>
<tr>
<td>T-&gt;Draw(&quot;x&gt;&gt;h1&quot;);</td>
<td>Dump a root branch to a histogram</td>
</tr>
</tbody>
</table>
Thanks!