The data chain of an experiment

DATA STORAGE ⇒ DATA ANALYSIS

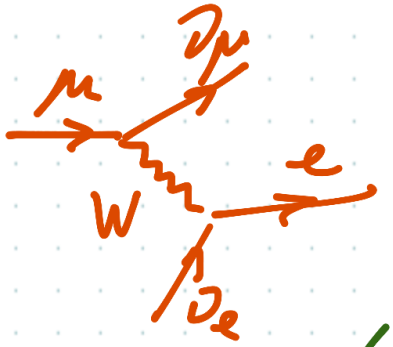
Fernando Barão (IST/LIP)

JUL. 2017

Our experiment

- Record data coming from muon decay

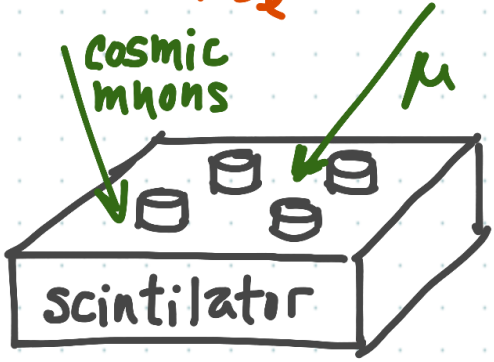
$$\begin{pmatrix} e \\ \nu_e \end{pmatrix} \quad \begin{pmatrix} \mu \\ \nu_\mu \end{pmatrix} \quad \begin{pmatrix} \tau \\ \nu_\tau \end{pmatrix}$$



$m_e = 0.511$ MeV/c²

$m_\mu = 106$ MeV/c²

$m_\tau = 1777$ MeV/c²



Physics of muon decay

- Energy and angular distributions of daughter electrons (positrons)

$\mu^{\pm} \to e^{\pm} + 2\nu$

$$\frac{d^2N}{d(\cos\alpha)\,dx} = \left[2x^2(3-2x)\right]\left[1 - \frac{1-2x}{3-2x}\cos\alpha\right]$$

$$x = \frac{2E_e}{m_\mu}$$

- What data to record?
  - run number → set of events
  - event number
  - muon decay time
  - electron energy
  - electron direction
  - electron velocity
- Let's use the monte-carlo technique to simulate our experiment
  - muon decay time measured with 1% precision
  - electron energy measured at 3%

Monte - Carlo simulation

- Generate the polar angle of electrons

$$\frac{dN}{d(\cos\alpha)} = \int_0^1 \frac{d^2N}{d(\cos\alpha)\,dx}\,dx \propto \left(1 + \frac{1}{3}\cos\alpha\right)$$

$\alpha \in [0, \pi]$

- Generate the electron energy

$$\frac{dN}{dE_e} = \int_{-1}^{1} \frac{d^2N}{d(\cos\alpha)\,dx}\,d(\cos\alpha) \propto 2x^2(3-2x)$$

$x \in [0,1]$

$$E_e^{max} = \frac{m_\mu}{2}$$

Technically:

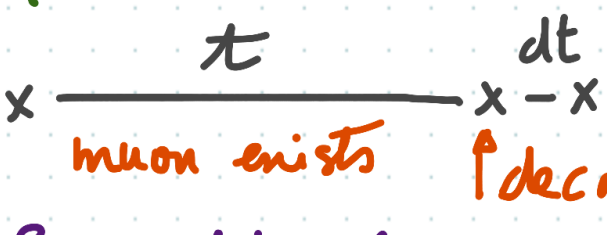
- Define TF1 objects and use the method GetRandom

```
TF1 *fE = new TF1("fE", "  ", 0., 1.);
                          ↑ function express.
double E = fE->GetRandom();
```

- Introduce the uncertainty of the measurement (3%)

```
double sigma = 0.03 * E;
double E_m = E + gRandom->Gaus(sigma)
```

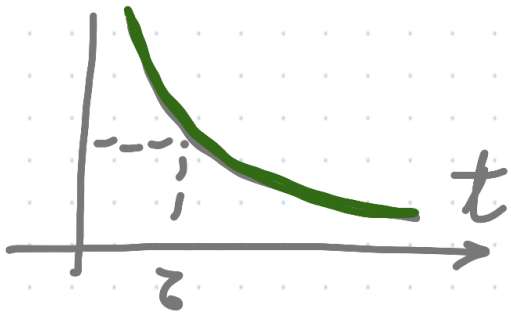
- Generate the muon decay time



Probability of a muon decaying after a time t

$$dP = \exp\left(-\frac{t}{\tau}\right) \cdot \frac{dt}{\tau}$$

$$\frac{dP}{dt} = \frac{1}{\tau} \exp\left(-t/\tau\right)$$




```
double tau = 2.1969 E-8; // secs
double t = gRandom->Exp(tau);
```

Storing data using ROOT

- We use TFile object (from ROOT) to store information

```
TFile *f = new TFile("muon.root","RECREATE");
```

↑ file name

- We use TTree object to store a colection of identical objects → our data events

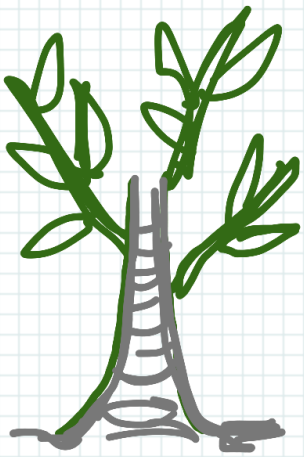
```
// docs
// https://root.cern.ch/doc/master/classTTree.html
//
```

```
TTree *tree = new TTree("MUON","muon decay tree");
```

↑ tree name

The muon event variables to store

```
muon event: tree variables
--------------------------

double muonTime; // muon decay time
double muonTimeMeas; // measurement

double electronEnergy; // electron energy
double electronEnergyMeas; // electron energy meas

double electronAlpha; // angle wrt muon
TVector3 electronDir; // electron direction
```



tree

tree → branchs → leafs

```
// ... define variables
Float_t muT, muTm, eE, eEm, eA;
TVector3* edir = new TVector3();

// ... set tree branchs/leafs
tree->Branch("muonTime",&muT,"muonTime/F"); //branch name, address, leaflist
tree->Branch("muonTimeMeas",&muTm,"muonTimeMeas/F");
tree->Branch("electronEnergy",&eE,"electronEnergy/F");
tree->Branch("electronEnergyMeas",&eEm,"electronEnergymMeas/F");
tree->Branch("electronAlpha",&eA,"electronAlpha/F");
tree->Branch("edir","TVector3",&edir);
```

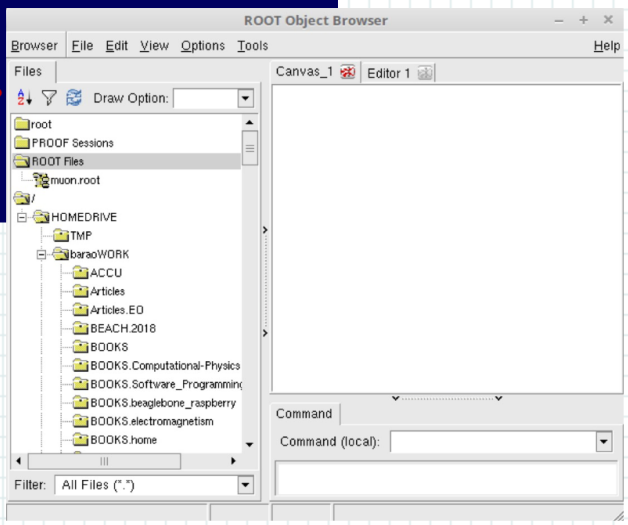
Generate events and store in the tree

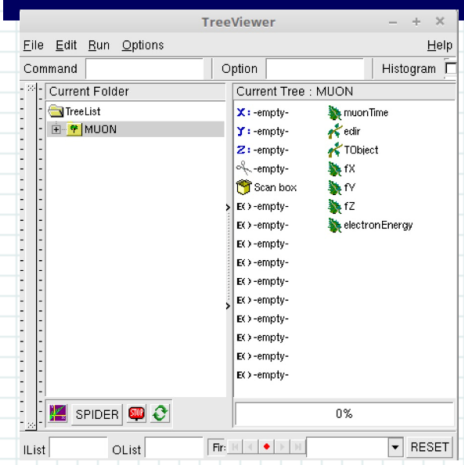
```
int Nevts = 1000;
for (int i =0; i<Nevts; ++i) {
  // muon decay time
  ...

  // electron direction (assume muon along z axis)
  float coseA = ...;
  eA = TMath::RadToDeg()*acos(coseA);
  float phi = gRandom(360.);
  float sinphi = sin(TMath::DegToRad()*phi);
  edir->SetXYZ( sqrt(1-coseA*coseA)*sqrt(1-sinphi*sinphi),
                sqrt(1-coseA*coseA)*sinphi,
                coseA);

  // electron energy
  ...

  // fill tree with events
  tree->Fill();
}
```

```
// close file
tree->Print();
f->Write();
f->Close();
```

Event data analysis : tree viewer

```
// fetch tree
TFile *f = new TFile("muon.root");
TTree *tree = (TTree*)f->Get("MUON");

// browse tree
new TBrowser();
tree->StartViewer();
```





Event data analysis : loop on events

- Open root file
- Retrieve tree
- Set tree variables
- book histos
- loop on tree entries

```
// set tree variables
Float_t mT, mTm, eE, eEm, eA;
TVector3 edir;

// set addresses where to put tree entries
tree->ResetBranchAddresses();
tree->SetBranchAddress("muonTime",&mT);
tree->SetBranchAddress("muonTimeMeas",&mTm);
...
tree->SetBranchStatus("*",1); //activate all branches

// histograms
TH1F *h = new TH1F("h","muon time (musecs)", 100, 0., 20.);

// loop on entries
Long64_t nentries = tree->GetEntries();
for (Long64_t i=0; i<nentries; ++i) {
  if (tree->GetEntry(i)<=0) break;
  h->Fill(mT);
}
```

Optimize tree reading

```
//read only the muonTime branch for all entries
TH1F *h = new TH1F("h","muon time (musecs)", 100, 0., 20.);
TBranch *b_muonTime = tree->GetBranch("muonTime");
b_muonTime->SetAddress(&mT);
Long64_t nentries = tree->GetEntries();
for (Long64_t i=0;i<nentries;i++) {
  b_muonTime->GetEntry(i);
  h->Fill(mT);
}
```

Event data analysis: treeDraw

```
// fetch tree
TFile *f = new TFile("muon.root");
TTree *tree = (TTree*)f->Get("MUON");

// simple analysis: htemp histo is created
tree->Draw("muonTime"); //histo 1D
tree->Draw("muonTime:muonTimeMeas"); //scatter plot

// making selection with Draw method
tree->Draw("electronEnergy","muonTime<4.");

// retrieve temporary histogram
TH1F *htemp = (TH1F*)gPad->GetPrimitive("htemp");

// pipe the result of the TTree::Draw into a histogram
tree->Draw("electronEnergy>>hnew(40,0.,20.)","muonTime<4.");
// get hnew from the current directory
TH1F *hnew = (TH1F*)gDirectory->Get("hnew");
```

Scan tree contents

```
root[] MyTree->Scan("*");
```

will print all the variable of the tree.

Specific variables of the tree can be explicit selected by list them in column separated list:

```
root[] MyTree->Scan("var1:var2:var3");
```

will print the values of `var1` , `var2` and `var3` . A selection can be applied in the second argument:

```
root[] MyTree->Scan("var1:var2:var3","var1==0");
```

will print the values of `var1` , `var2` and `var3` for the entries where var1 is exactly 0.

`TTree::Scan` returns the number of entries passing the selection. By default 50 rows are shown before `TTree::Scan` pauses and ask you to press the Enter key to see the next 50 rows. You can change the default number of rows to be shown before <CR> via `mytree->SetScanfield(maxrows)` where maxrows is 50 by default. If maxrows is set to 0 all rows of the **`Tree`** are shown. This option is interesting when dumping the contents of a Tree to an ascii file, eg from the command line: