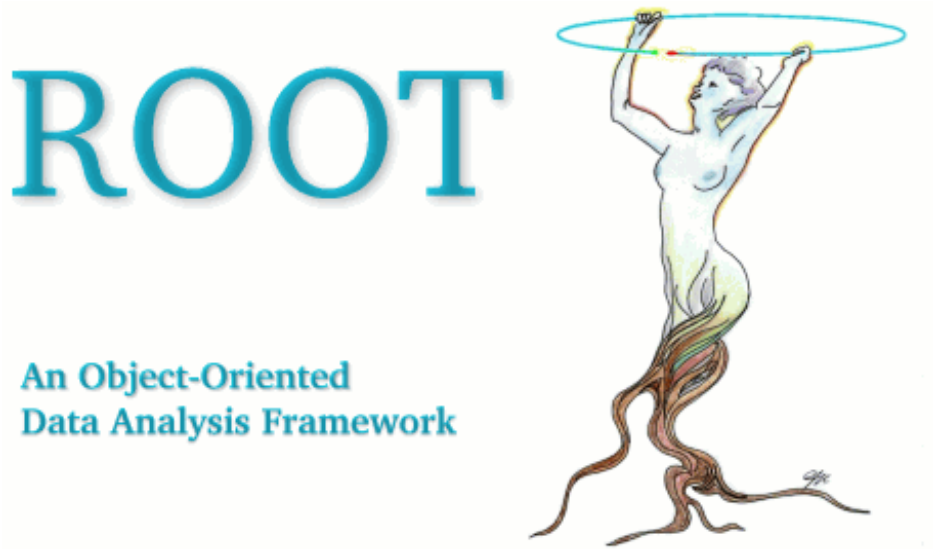
ROOT 5



http://root.cern.ch/

Random generators

TRandom
basic Random number generator class (periodicity = 10**8).
The following basic Random generators are provided:
-Exp(tau)
-Integer(imax)
-Gaus(mean,sigma)
-Rndm()
-Uniform(x1) -Uniform(x1,x2)
-Landau(mpv,sigma)
-Poisson(mean)
-Binomial(ntot,prob)

```
for (i=0;i<N;i++)  {
        x = gRandom->Gaus(0,1);
        h2->Fill(x) }
```

Κατανομή τυχαίων αριθμών με βαση μια συνάρτηση 1-d, 2-d ή 3-d

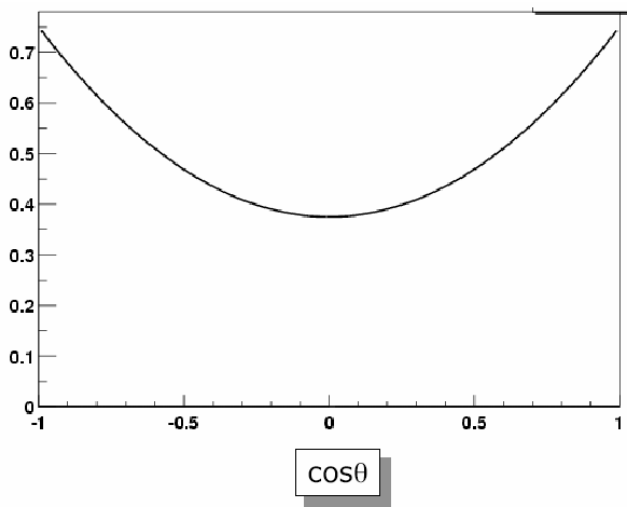
```
TF1 *f1 = new TF1("f1","abs(sin(x)/x)*sqrt(x)",0,10);
double r = f1->GetRandom();
```

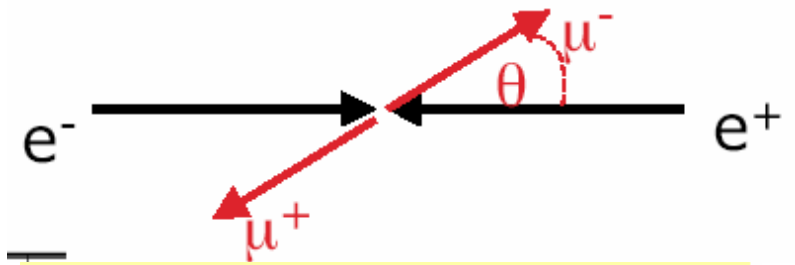
void Rannor(Float_t &a, Float_t &b) Return 2 numbers distributed following a gaussian with mean=0 and sigma=1

void SetSeed(UInt_t seed) Set the random generator seed if seed is zero, the seed is set to the current machine clock

Άσκηση

- Κατανομή της πολικής γωνίας θ του μιονίου στην σκέδαση ee→μμ

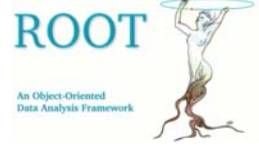




$$\frac{1}{\sigma_{tot}}\frac{d\sigma}{d\cos\theta} = \equiv f(\cos\theta) = \frac{3}{8}\left[1+(\cos\theta)^2\right]$$

Δημιουργείστε μια κατανομή 100.000 γεγονότων που ακολουθούν την παραπάνω κατανομή.

Άσκηση (συνέχεια)



- Κατανομή της αζιμουθιακής γωνίας φ του μιονίου στην σκέδαση ee→μμ είναι ομοιόμορφη (Uniform)

- Δημιουργείστε μια κατανομή 100.000 γεγονότων που ακολουθούν την παραπάνω κατανομή

- Δώστε σε ιστόγραμμα 2D τις κατανομές των γωνιών φ και cosθ

Random generators

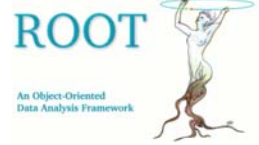
ΑΣΚΗΣΗ

(δημιουργείστε hprofile.c)

- Δημιουργείστε 40.000 σωματίδια για τα οποία ισχύει px, py κατανομές Gaus και pt=px*px+py*py.
- Σχέση px-py, px-pz (TH2F και TProfile)

```
hprof = new TProfile("hprof","Profile of pz versus px",100,-4,4,0,20);
```

ROOT I/O system



Αρχεία ROOT

Ένα αρχείο ROOT είναι όπως το σύστημα αρχείων του Unix (directories, levels …)

Κάθε κατάλογος συσχετίζεται με μια λίστα κλειδιών (KEYS) που χαρακτηρίζουν τα αντικείμενα που διατηρούνται στη μνήμη μέχρι να κλείσει το αρχείο.

hsimple.root

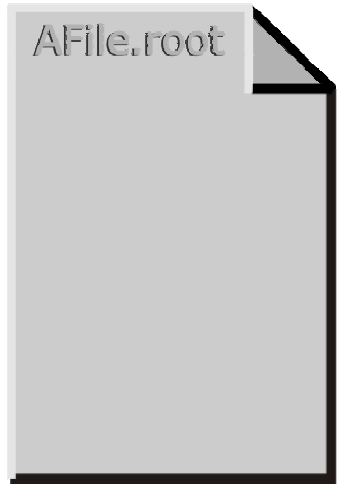
Χρησιμοποιώντας Browser

```
new TBrowser
```

hsimple.root

```
TFile f1("hsimple.root")
f1.Map()
f1.ls()
f.GetListOfKeys()->Print()
```

ROOT Files (TFile)



- Όταν ανοίγει ένα αρχείο ROOT

γίνεται ο τρέχων κατάλογος.

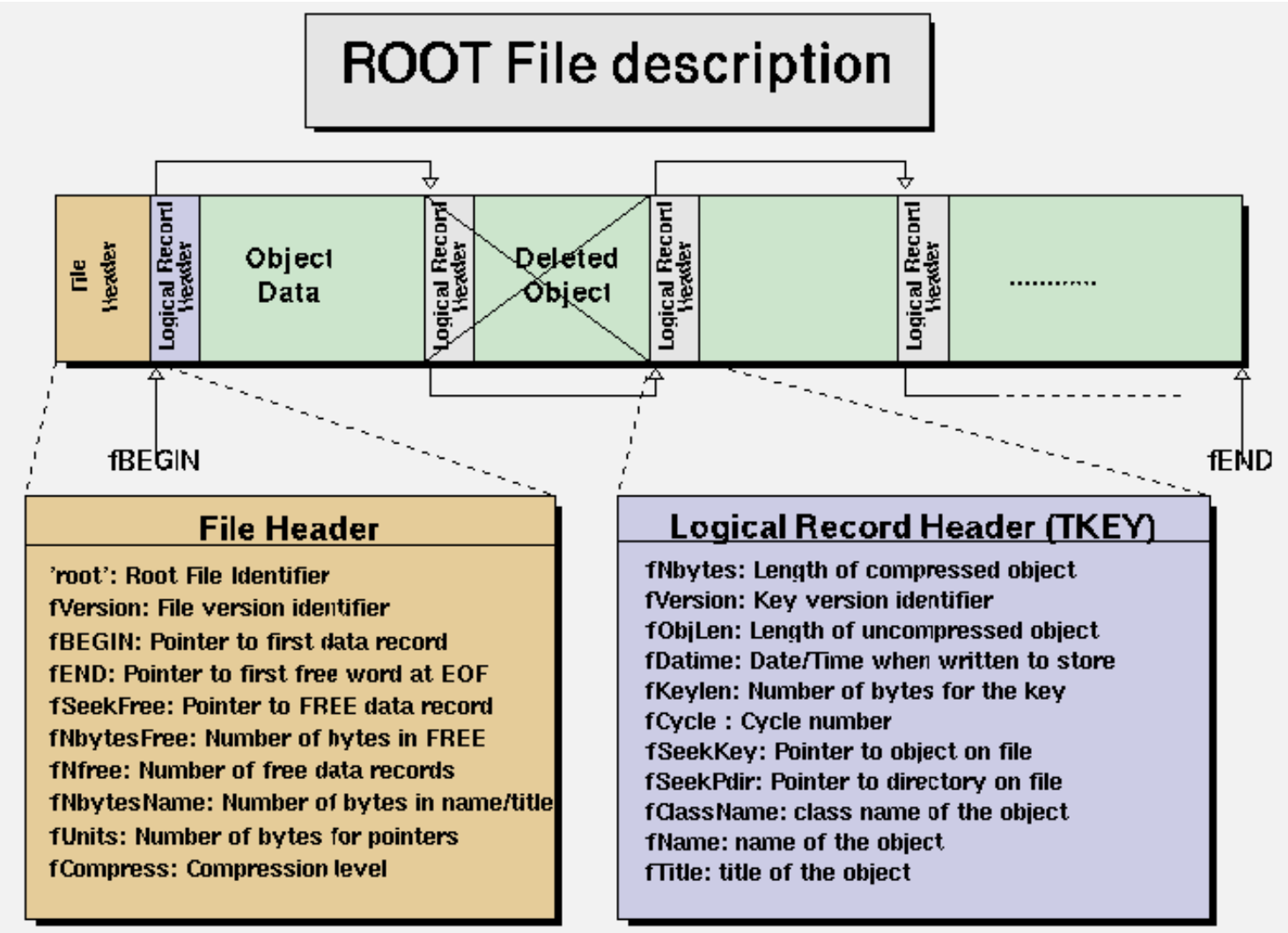
- Ιστογράμματα και δέντρα σώζονται

αυτόματα στο αρχείο.

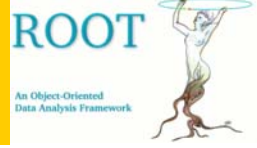
- Όταν κλείνει το αρχείο τα αντικείμενα ιστογράμματα και δέντρα που σχετίζονται με το αρχείο σβήνονται.
- Κάθε αντικείμενο που προέρχεται από TObject μπορεί να γραφεί σε αρχείο ROOT. Πρέπει να δοθεί «ρητή» εντολή.

π.χ. C1->Write();

Αρχεία ROOT (δομή)



Αρχεία ROOT



File Header Information		
Byte	**Value Name**	**Description**
1 -> 4	`"root"`	Root file identifier
5 -> 8	`fVersion`	File format version
9 -> 12	`fBEGIN`	Pointer to first data record
13 -> 16	`fEND`	Pointer to first free word at the EOF
17 -> 20	`fSeekFree`	Pointer to FREE data record
21 -> 24	`fNbytesFree`	Number of bytes in FREE data record
25 -> 28	`nfree`	Number of free data records
29 -> 32	`fNbytesName`	Number of bytes in `TNamed` at creation time
33 -> 33	`fUnits`	Number of bytes for file pointers
34 -> 37	`fCompress`	Zip compression level

Αρχεία ROOT

Record Information (TKEY)		
Byte	**Value Name**	**Description**
1 -> 4	`Nbytes`	Length of compressed object (in bytes)
5 -> 6	`Version`	TKey version identifier
7 -> 10	`ObjLen`	Length of uncompressed object
11 -> 14	`Datime`	Date and time when object was written to file
15 -> 16	`KeyLen`	Length of the key structure (in bytes)
17 -> 18	`Cycle`	Cycle of key
19 -> 22	`SeekKey`	Pointer to record itself (consistency check)
23 -> 26	`SeekPdir`	Pointer to directory header
27	`lname`	Number of bytes in the class name
28->…	`ClassName`	Object Class Name
…->…	`lname`	Number of bytes in the object name
…->…	`Name`	`lName` bytes with the name of the object
…->…	`lTitle`	Number of bytes in the object title
…->…	`Title`	Title of the object
----->	`DATA`	Data bytes associated to the object

Αρχεία ROOT

Εγγραφή ενός αντικειμένου σε αρχείο

TObject::Write()

Εκτελούνται οι παρακάτω διαδικασίες :

- **Δημιουργία ενός TKey αντικειμένου στον τρέχοντα κατάλογο**
- **Το αντικείμενο TKey δημιουργεί ένα TBuffer**
- **Το TBuffer γεμίζει μέσω της συνάρτησης Streamer()**
- **Δημιουργείτε ένας δεύτερος buffer για την συμπίεση**
- **Ελέγχεται η λίστα TFree για τα ελεύθερα block στο αρχείο**
- **Ο buffer γράφεται στο αρχείο**

Ανάγνωση αντικειμένου από αρχείο στη μνήμη

TKey::Read()

Εκτελούνται οι παρακάτω διαδικασίες :

- **Αναζητείται το αντικείμενο με βάση το κλειδί στον τρέχοντα κατάλογο**
- **Δημιουργείτε ένας buffer(s) για το αντικείμενο**
- **Με βάση το όνομα της Class του αντικειμένου (περιέχεται στο TKey) καλείται η TClass::New()**
- **Καλείται η συνάρτηση Streamer()**

Συμπίεση

Όλα τα δεδομένα συμπιέζονται πριν την εγγραφή σε αρχείο.

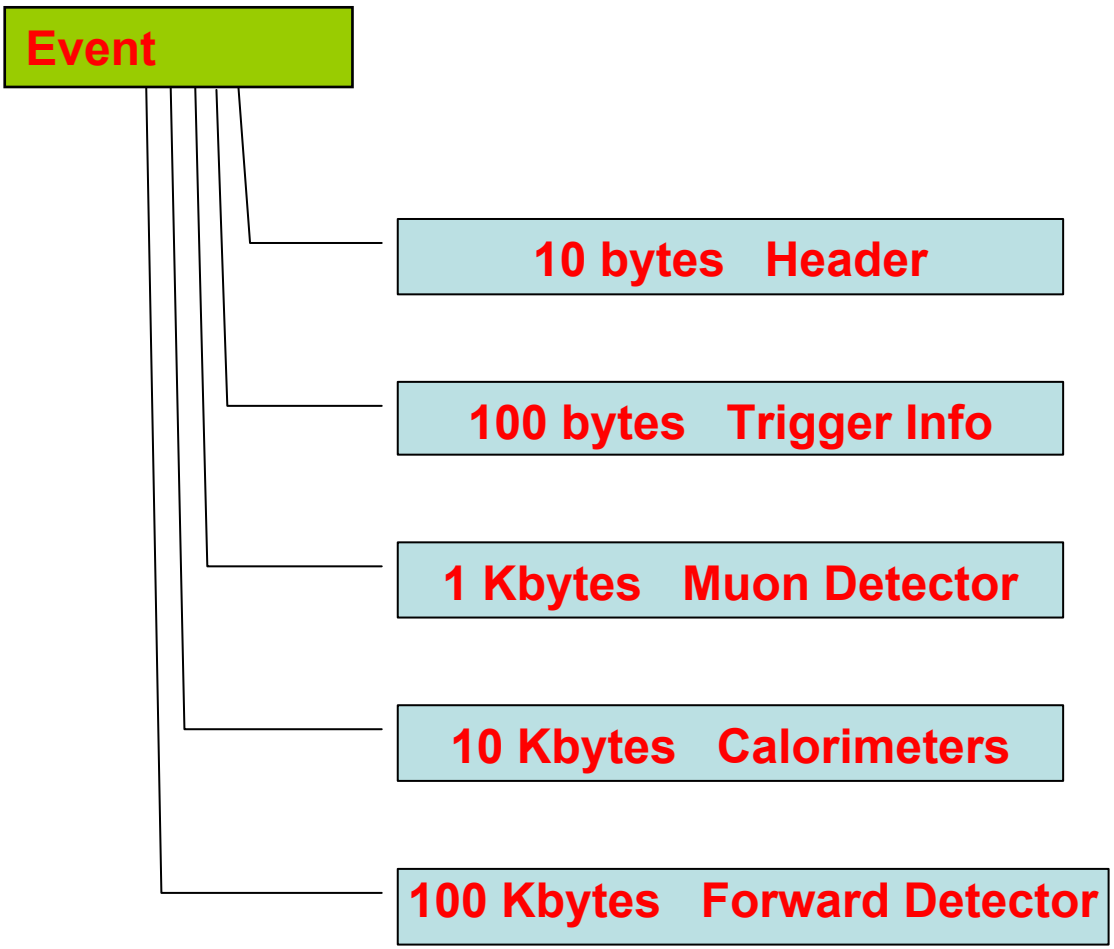
Χρησιμοποιείται ο αλγόριθμος gzip (9 levels)

Το επίπεδο συμπίεσης είναι 1

TFile::SetCompressionLevel()

level=0 σημαίνει χωρίς συμπίεση.

Event format in HEP



Επιτυχής για

Σειριακά αρχεία

Μεγάλο αριθμό δεδομένων

Μή αποδοτικό

Όταν το ενδιαφέρον εστιάζεται σε ένα μικρό υποσύνολο δεδομένων

ή σε υποσύνολο των ιδιοτήτων του γεγονότος

Ntuples - Trees

Δύο κύριες τάξεις

TTree

Μπορεί να αποθηκεύσει οποιαδήποτε δομή δεδομένων

Οποιαδήποτε μορφή δεδομένων

Πεδία μπορούν να ομαδοποιηθούν ή να αποθηκευτούν χωριστά

TNtuple

Απλούστερη μορφή Δέντρου

Μπορεί να θεωρηθεί ένας πίνακας αριθμών 2D

Επιτρέπει την εφαρμογή cuts σε μια ή περισσότερες μεταβλητές για την εμφάνιση (plotting) μιας άλλης.

Ntuples - Trees

Είναι κατάλληλες για την αποθήκευση / επεξεργασία δεδομένων με μορφή "Γεγονότων"

(Επαναλαμβανόμενη δομή δεδομένων)

```
Event 1
float vis_energy;
int ntracks, nclusters;
float px[max_tracks], py[...
```

```
Event 2
float vis_energy;
int ntracks, nclusters;
float px[max_tracks], py[...
```

. . .

Στο profile.c προσθέστε ntuple όπου κάθε γεγονός θα περιέχει τις px, py, pz.

ntuple = new TNtuple("ntuple", "Demo ntuple", "px:py:pz");

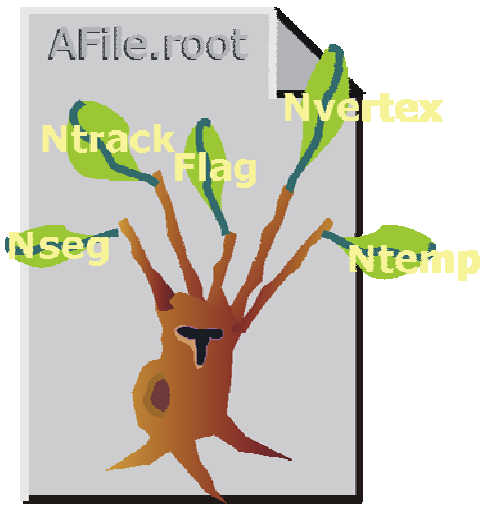
f = new TFile("hprof.root", "RECREATE");

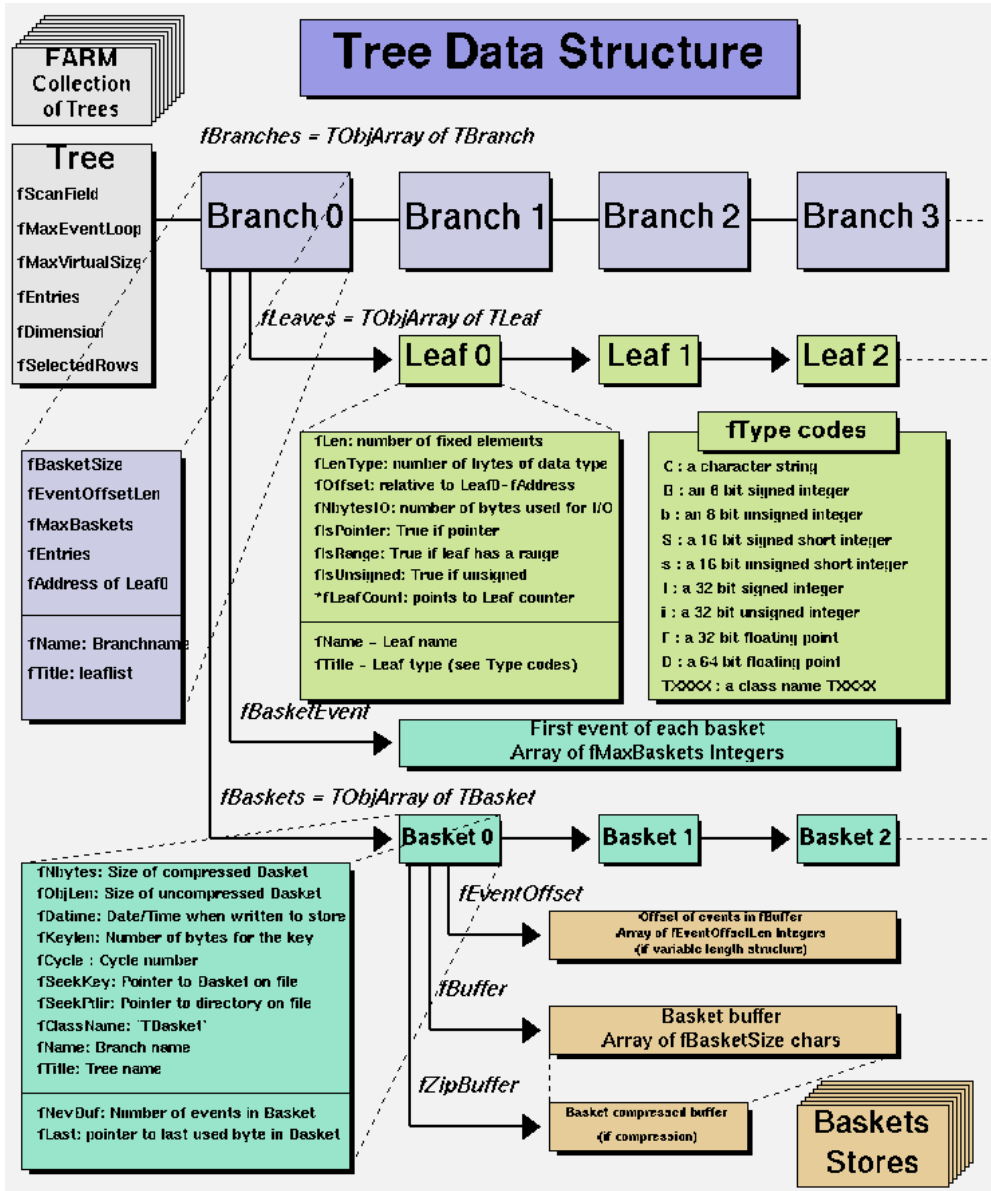
f ->Write();

Δέντρα ROOT (trees)

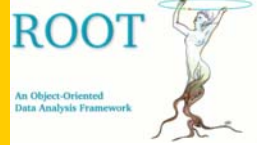
Ένα Δέντρο (Tree) αποτελείται από Κλάδους (Branch)

Και ένας κλάδος περιγράφεται από τα φύλλα (leaves) του !!!





Παράδειγμα

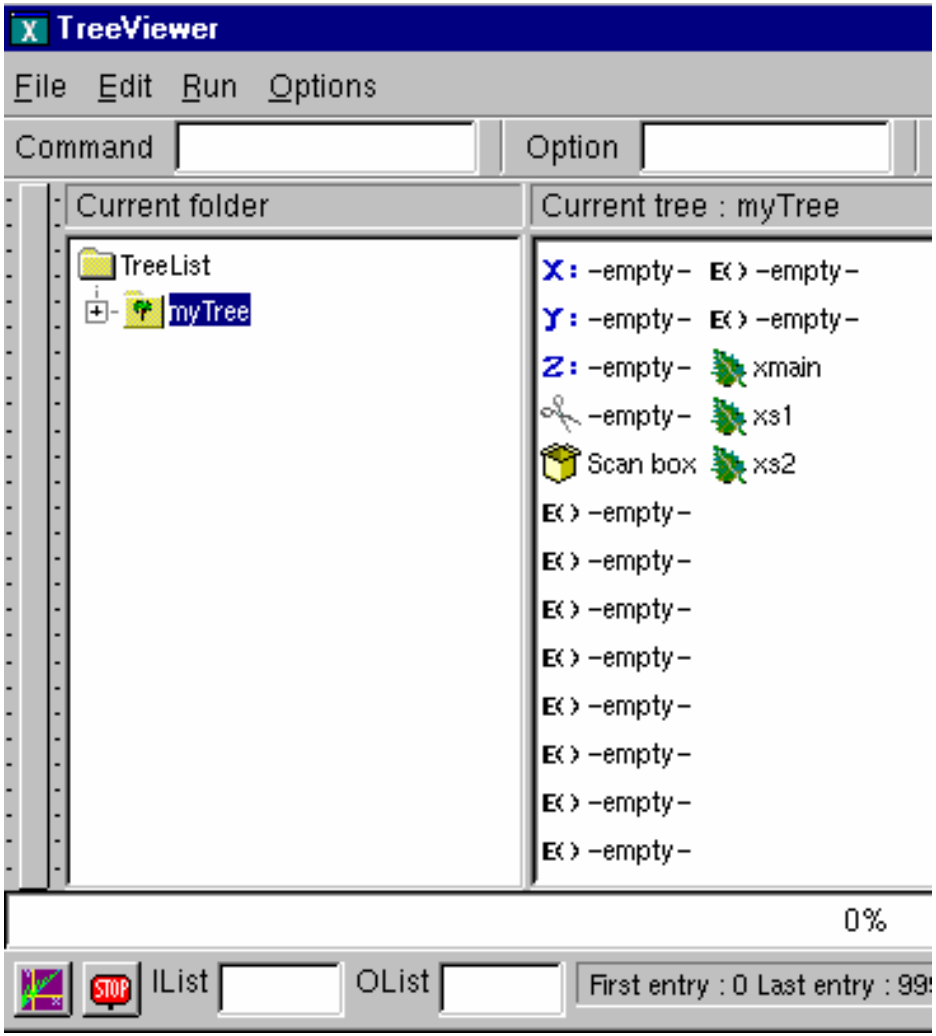


cernstaff.dat αρχείο κειμένου με πληροφορίες

cernstaff.C

cernbuild.C

The Tree Viewer



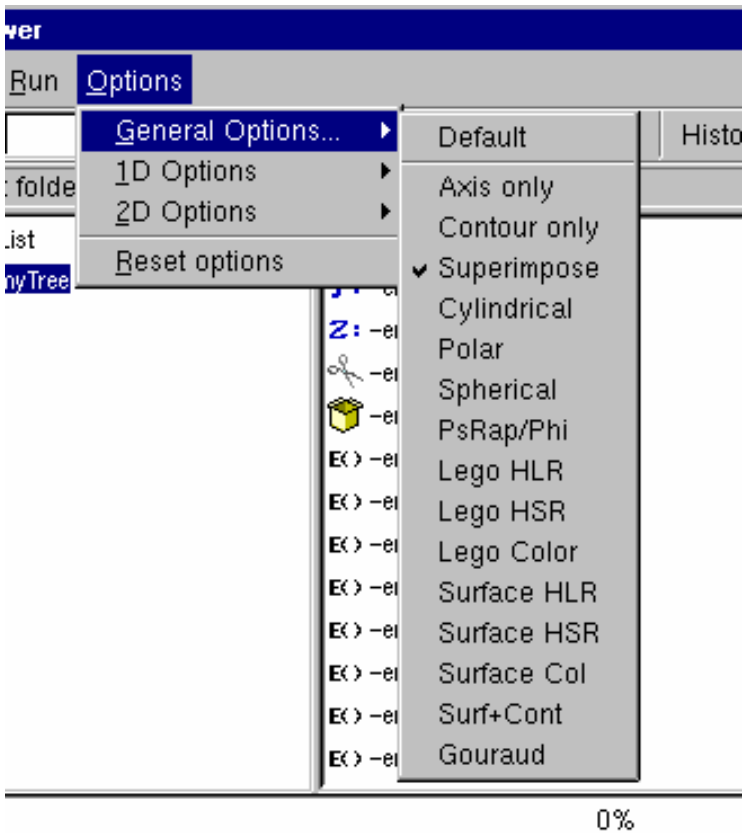
root[] myTree->StartViewer()

Contents:

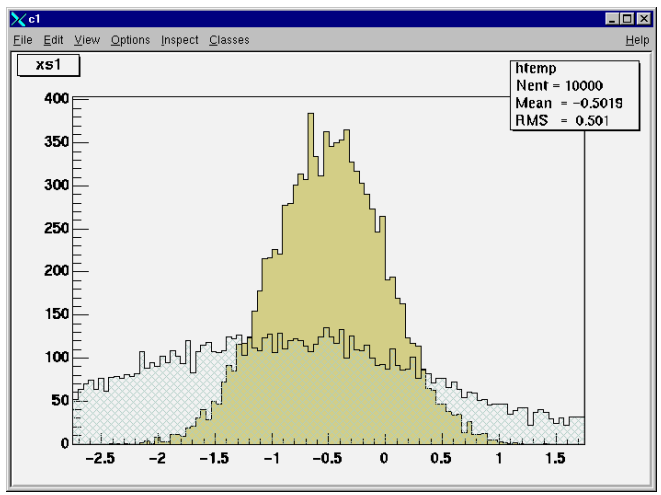
- XYZ
- Cut
- Scan
- Expressions
- Variables (leaves)
- Draw option
- Recording commands
- Histogram
- Range Selector
- IList/OList

tree1.c

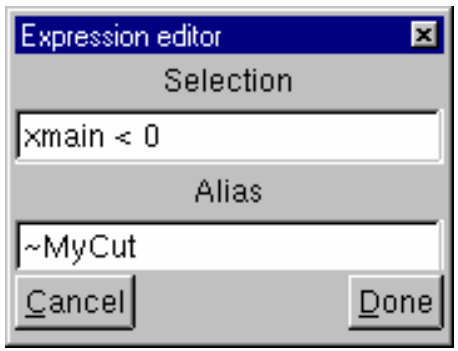
Drawing Two Histograms



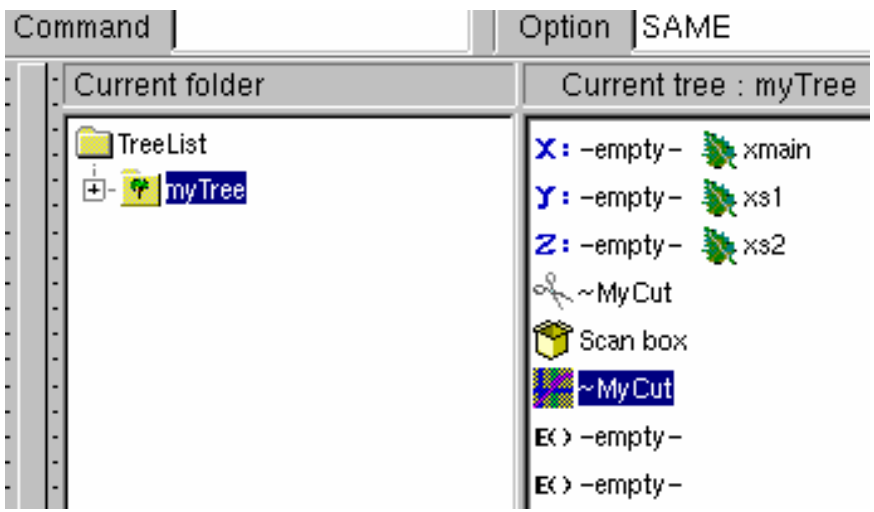
- Double click σε μια μεταβλητή
- Επιλέξτε Superimpose option
- Double click σε μια δεύτερη μεταβλητή

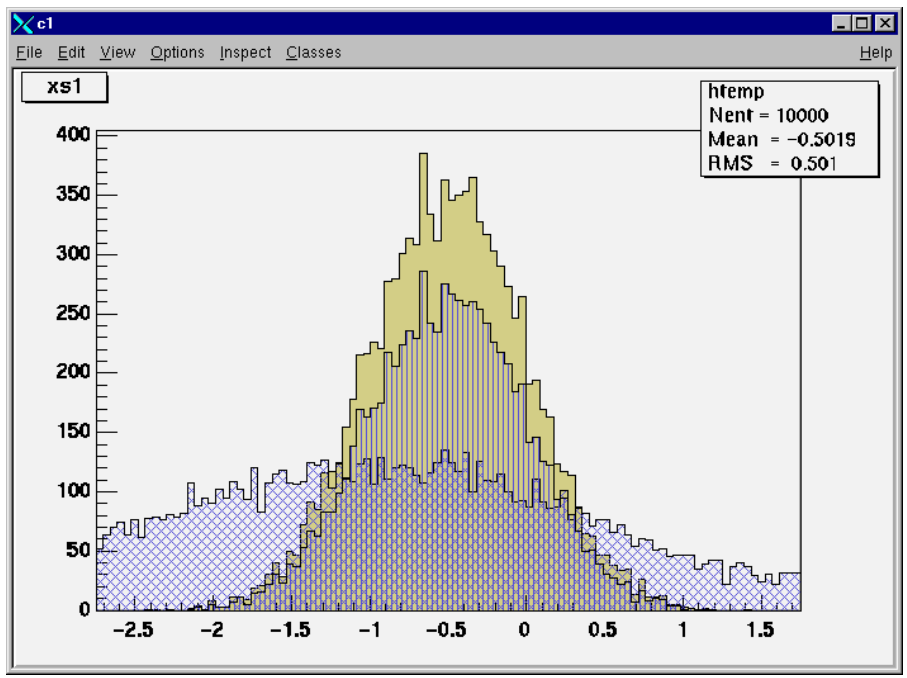


Adding Weight

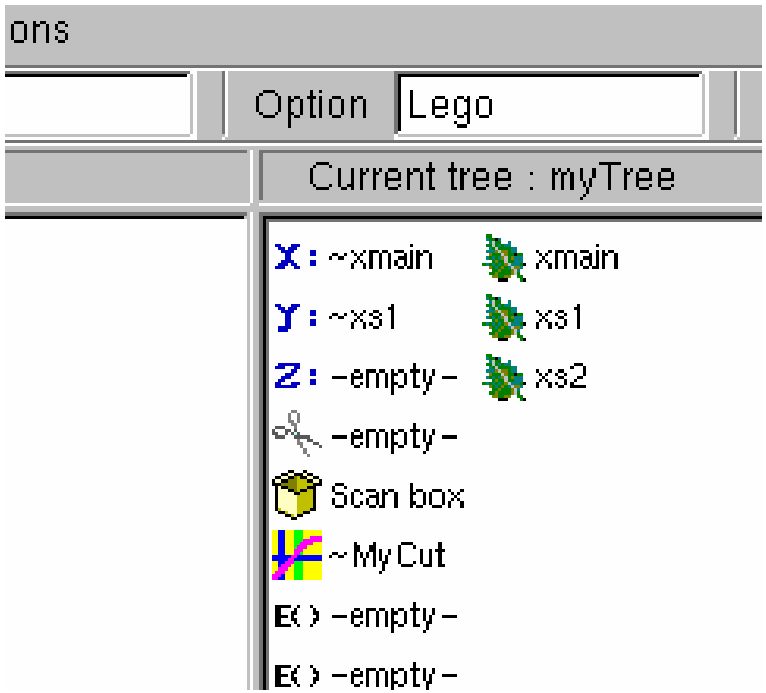


Add a Weight: xmain < 0

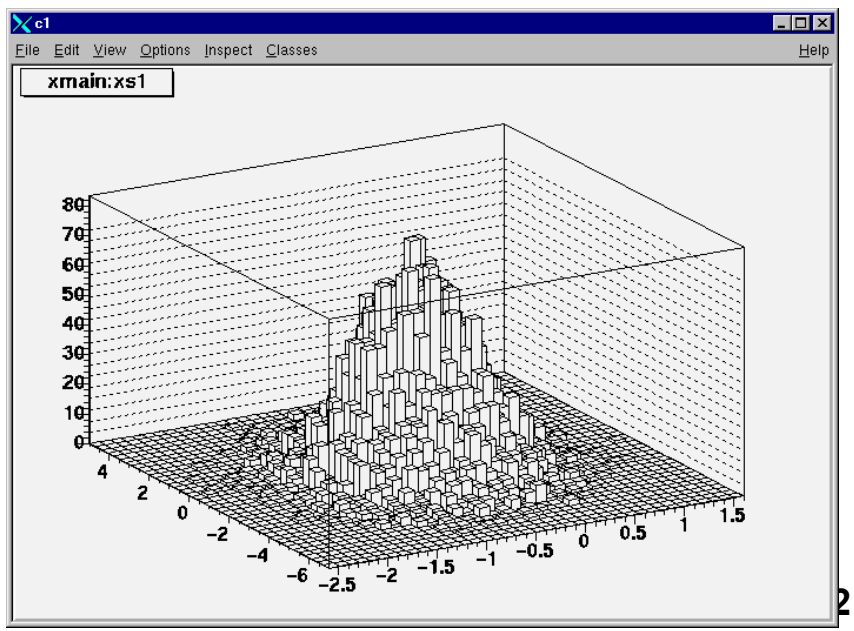




2D Lego Plots

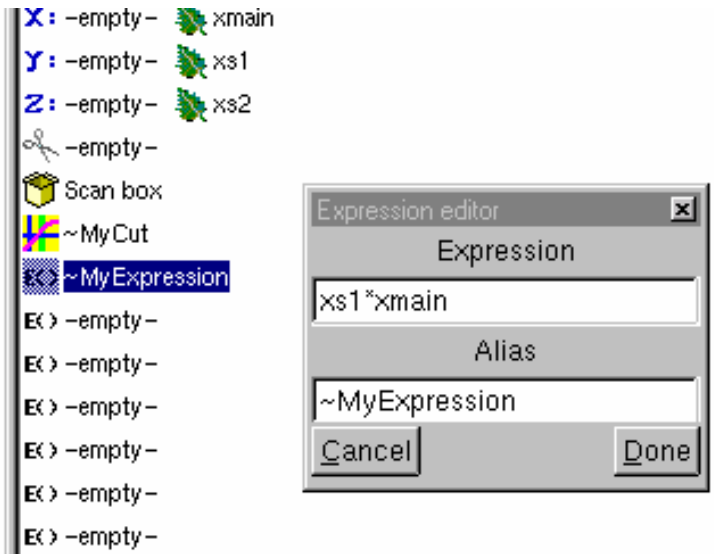


- Draw xmain vs. xs1
- Rotating the lego plot

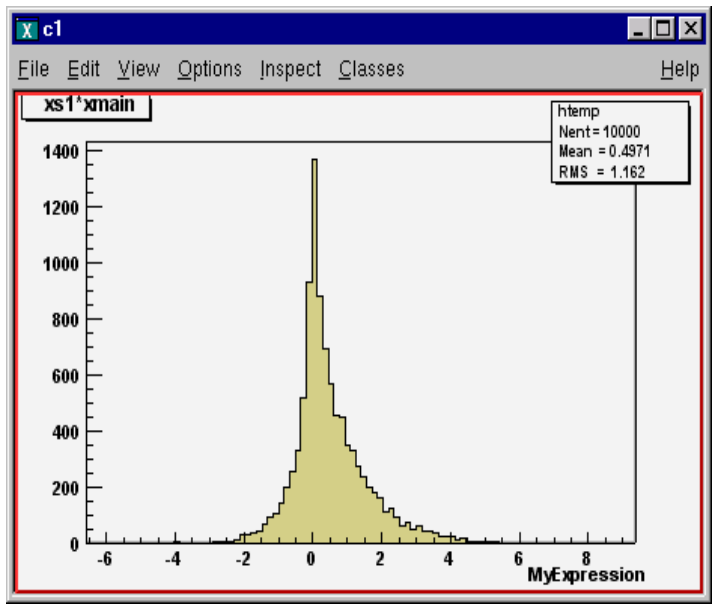


Creating a variable

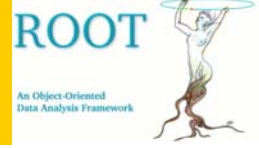
1. New Expression



2. Set the name and expression
3. Double click to draw it



More Features



Record:

1) Command is recorded in history file ($HOME/root_hist)
2) Is printed on the command line

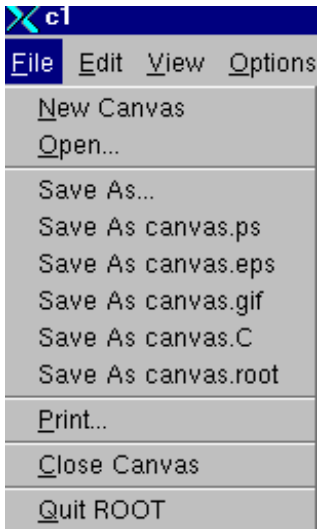
Scan:

1) Drag and drop variables to scan box
2) Redirect output to a file

Saving and recovering a Session:

1) Save the current cuts, expressions into treeviewer.C
2) Read session back

Saving the Canvas



- The save options:
  - Postscript
  - gif
  - ROOT file
  - Macro
- Save as a ROOT file:
  - Open the saved ROOT file (c1.root) and draw the canvas

```
root[2] > c1.Draw();
```

Read Tree

```
{   //read the Tree in hprof.root and fill two histograms
   //read the file and get the tree
 TFile *f = new TFile(“hprof.root");
 TTree *t1 = (TTree*)f->Get(“ntuple");
 Float_t px, py;
 pxBranch = t1->GetBranch("px");
 pxBranch->SetAddress(&px);
 ……..
//create two histograms
TH1F *hpx   = new TH1F("hpx","px distribution",100,-3,3);
TH2F *hpxpy = new TH2F("hpxpy","py vs px",30,-3,3,30,-3,3);

//read all entries and fill the histograms
Int_t nentries = (Int_t) t1->GetEntries();

for (Int_t i=0;i<nentries;i++) {     pxBranch->GetEntry(i);
                                     pyBranch->GetEntry(i);
                                     hpx->Fill(px);     hpxpy->Fill(px,py);  }
// Draw the histograms
TCanvas c;  c->Divide(2,1);  c->cd(1);
hpx->Draw();  c->cd(2);  hpxpy->Draw("lego");
 }
```

Read Tree (2)

```
TFile *f = new TFile("hprof.root");
ntuple->MakeClass("ana")
```

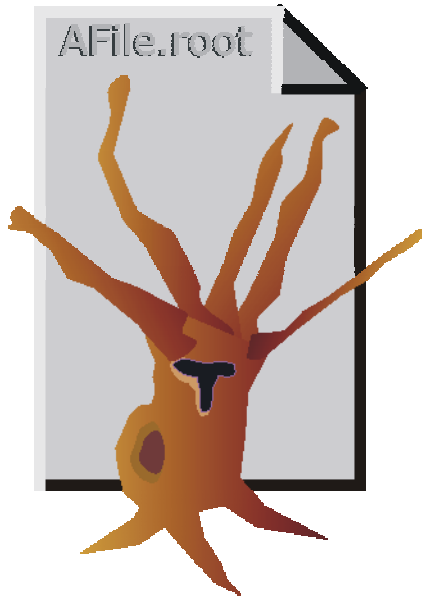
Δημιουργεί τα αρχεία ana.C και ana.h

Στη μέθοδο ana::Loop() μπορούμε να προσθέσουμε τον κώδικά μας…..
Για να τρέξει

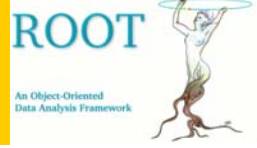
```
.L ana.C
ana   m   // δημιουργεί το object  m
m.Loop() // τρέχει  η μέθοδος Loop()
```

ROOT Trees (TTree)

- Αποθήκευση μεγάλου αριθμού δεδομένων.
- Ιεραρχία κλάδων και φύλλων (branches and leaves).
- Διάβασμα επιλεγμένων κλάδων .



Δημιουργία δένδρου



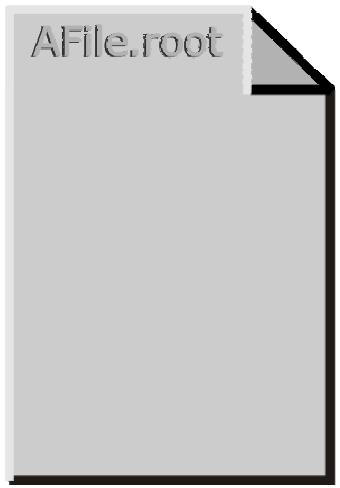
Βήματα:

1. Δημιουργία ενός TFile
2. Δημιουργία ενός TTree
3. Προσθήκη TBranch στο TTree
4. Γέμισμα του δένδρου tree
5. Εγγραφή σε αρχείο (Write the file)

Step 1: Create a TFile Object

TFile constructor

- file name (i.e. " AFile.root ")
- option: NEW, CREATE, RECREATE, UPDATE, or READ
- file title
- compression level 0-9, defaults to 1.

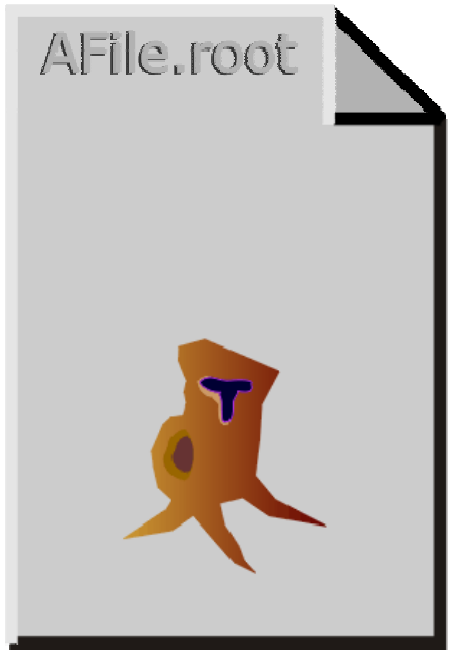



```
TFile *hfile = new
   TFile("AFile.root","RECREATE","Example");
```

Step 2: Create a TTree Object

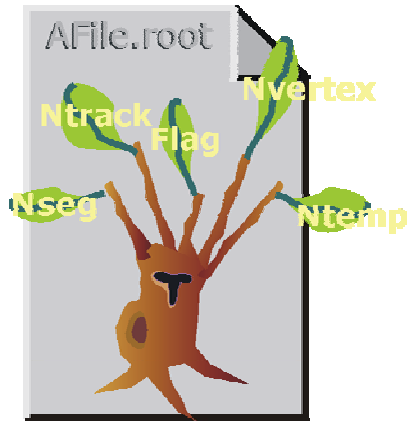
TTree Constructor:

- Tree Name (e.g. "myTree")
- Tree Title
- Maximum total size of buffers kept in memory when reading a TTree (defaults to 64 MB)




```
TTree *tree = new TTree("myTree","A ROOT tree");
```

Step 3: Adding a Branch

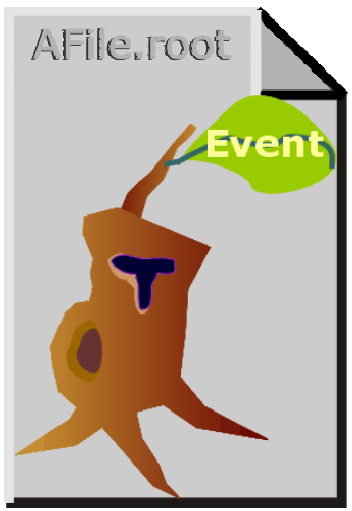


- Branch name
- Class name
- Address of the pointer to the Object
- Buffer size (default = 32,000)
- Split level (default = 99)

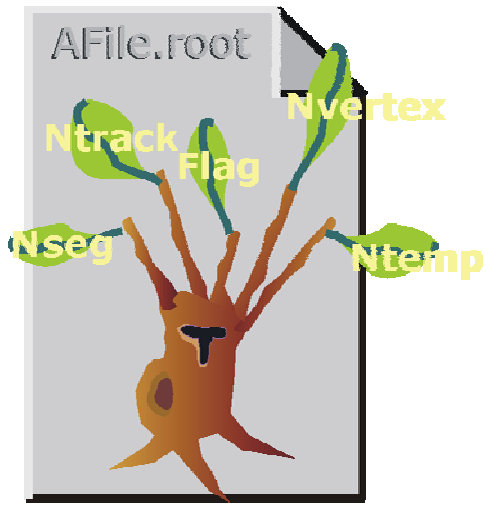
```
Event *event = new  Event();
myTree->Branch("EventBranch","Event",&event);
```

Splitting a Branch

Setting the split level (default = 99)



Split level = 0

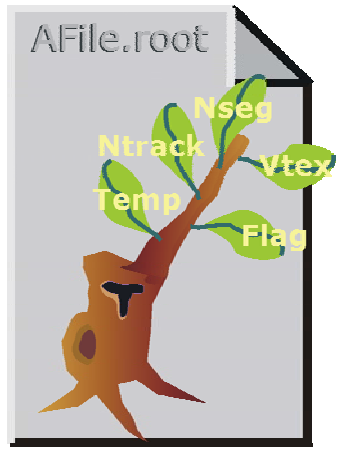


Split level = 99

Example:

```
tree->Branch("EvBr","Event",&ev,64000,0);
```

Branches with a List of Variables



- Branch name
- Address: the address of the first item of a structure.
- Leaflist: all variable names and types
- Order the variables according to their size

Example

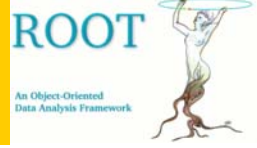
```
TBranch *b = tree->Branch ("Ev_Branch",&event,
      "ntrack/I:nseg:nvtex:flag/i:temp/F");
```

Adding Branches with a TClonesArray

- Branch name
- Address of a pointer to a TClonesArray
- Buffer size
- Split level


```
Example:
tree->Branch( "Track_B",&Track,64000);
```

List and Folder Branches

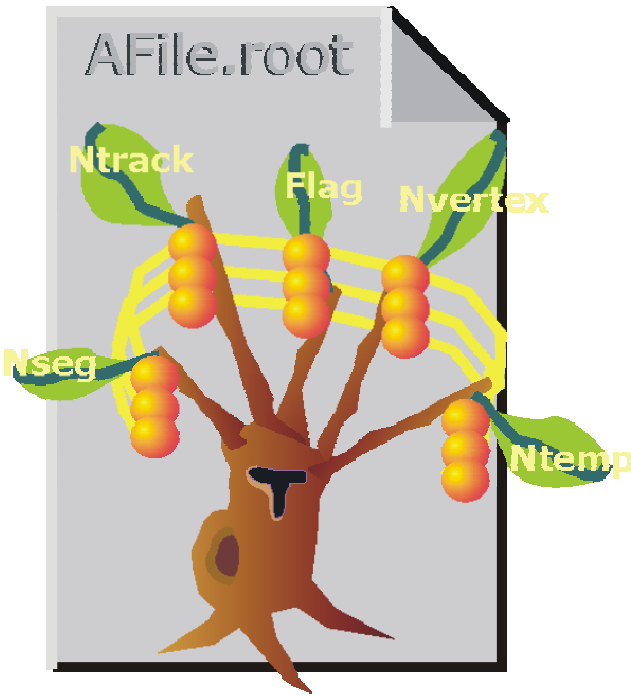


- Branch(TList *list, buffer, split)
  - Creates one branch for each list element
  - TObject
  - TClonesArray
  - Will add a split level parameter
- Branch("folder-name", buffer, split)
  - Creates one branch per folder

Step 4: Fill the Tree

- Create a for loop
- Assign values to the event object
- Call the Fill method for the tree

```
myTree->Fill()
```

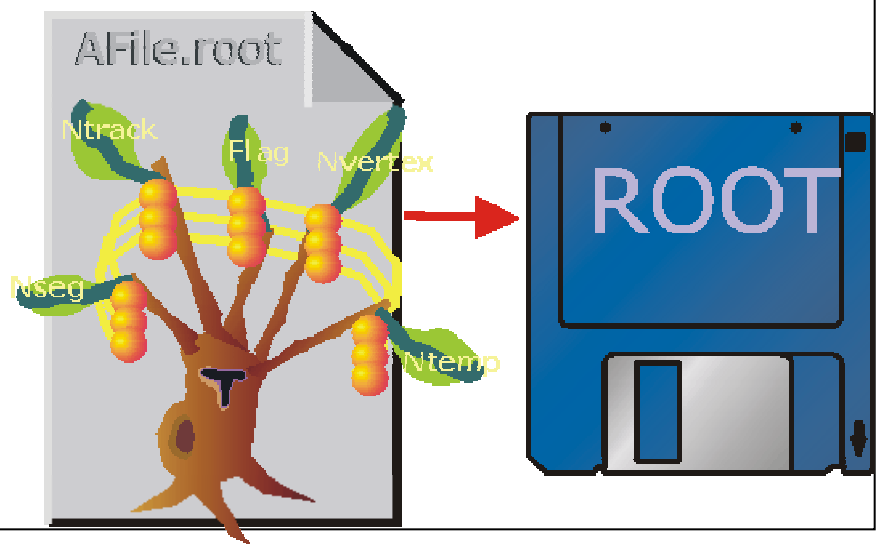


Step 5: Write the File

The TFile::Write()

- Writes Histograms and Trees
- Write is needed to write file header

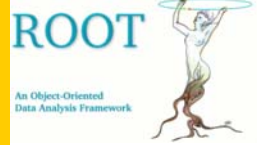
```
hfile->Write();
```



Reading a TTree

- How to Read a Tree
- Reading Simple variables
- Example: reading Selected Branches
- Example: reading an Object Branch
- Trees and Friends

Looking at the Tree



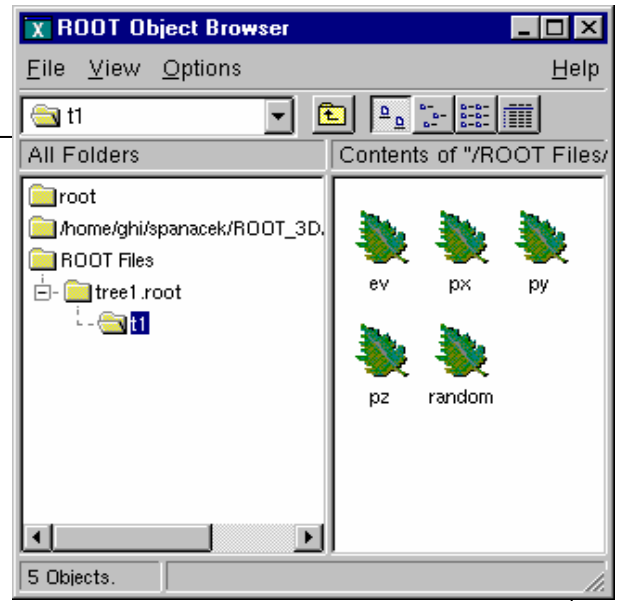
- TTree::Print() Shows the branches

```
TFile f("AFile.root")
myTree->Print(); > print.txt
```

- TTree::Scan("leaf":"leaf":....)

```
myTree->Scan("fNseg:fNtrack"); > scan.txt
myTree->Scan("fEventHdr.fDate:fNtrack");
```

How To Read TTree



$ROOTSYS/tutorials/**tree1.C**

Reading a simple tree

1. Open the TFile

```
TFile f("tree1.root")
```

2. Get the TTree

```
TTree * t1 =
   (TTree*)f.FindObject("t1")
```

How to Read A TTree ++

3. Create a variable to hold the data

```
Float_t px, py, pz;
```

4. Associate a branch with a variable:

`SetBranchAddress("name", address)`

```
t1->SetBranchAddress("px", &px)
t1->SetBranchAddress("py", &py)
t1->SetBranchAddress("pz", &pz)
```

GetEntry

5. Read one Entry in the TTree

```
t1->GetEntry(0) // first entry
```

root [20] px

(Float_t)(-1.10227906703948970e+00)

root [21] py

(Float_t)(-1.79938960075378420e+00)

root [22] pz

(Float_t)4.45282220840454100e+00

Looking at the Tree

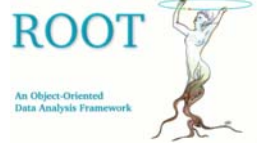
Εκτός του δένδρου έχουμε και δύο ιστογράμματα

```
//create two histograms
TH1F *hpx   = new TH1F("hpx","px distribution",100,-3,3);
TH2F *hpxpy = new TH2F("hpxpy","py vs px",30,-3,3,30,-3,3);
```

Βλέποντας στον TBrowser

Ανοίγουμε ξανά το tree1.root

Demo: Reading Branches



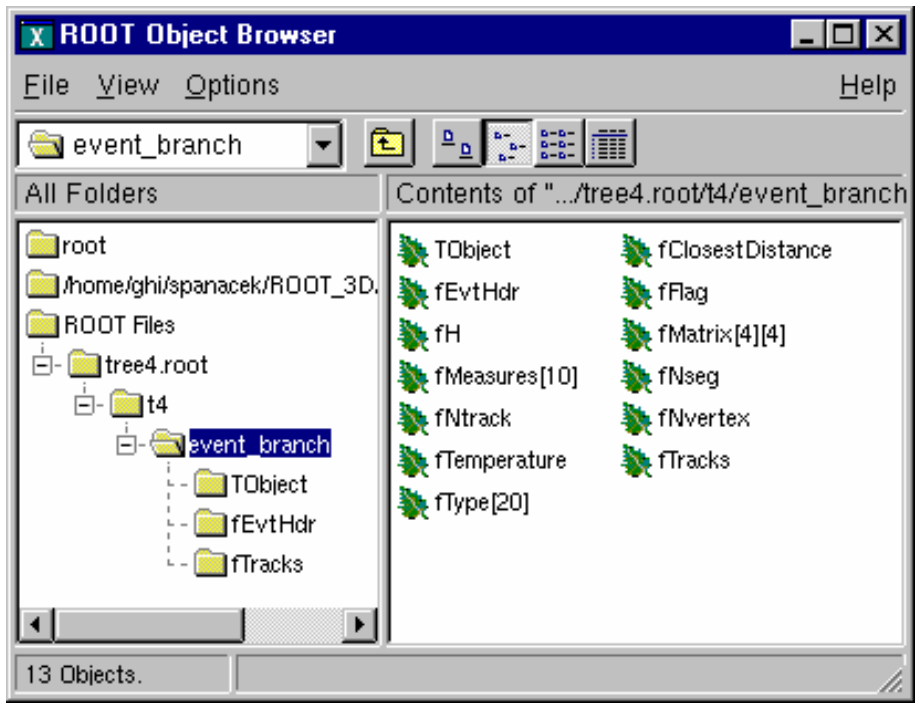
- Demo:readTree1.C
  - Read selected branches
  - Fill two histograms

Reading an Object Branch

Print the first entry with less than 587 tracks

Find the entry using the fNtrack sub-branch

Once found, read the entire entry



$ROOTSYS/tutorials/tree4.C