AISF

comitato locale di Perugia

AISF - COMITATO LOCALE DI PERUGIA

TREES
STORING DATA

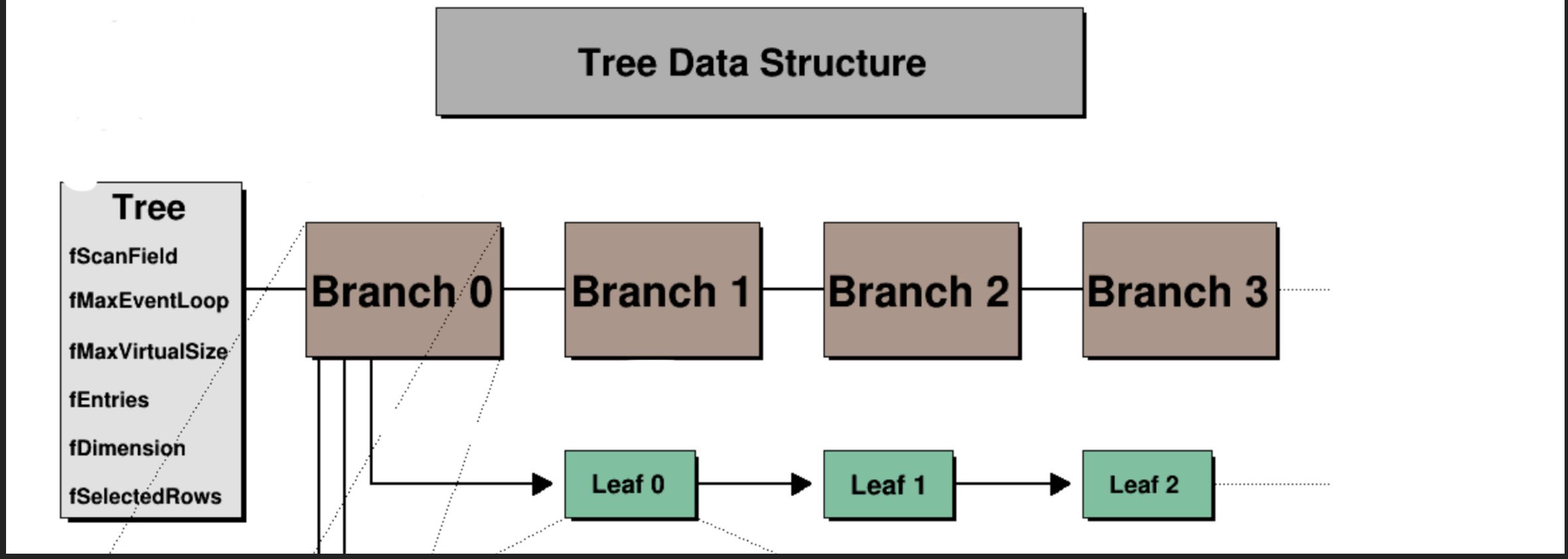
Intro

OLTRE AI TOOLS GIÀ VISTI PER UN ANALISI DATI DI PRIMO LIVELLO, ROOT FORNISCE UNA CLASSE ADATTA AD IMMAGAZZINARE GRANDI QUANTITÀ DI DATI. L'USO DELLA CLASSE TTREE PER L'APPUNTO OFFRE VANTAGGI NELLA COMPRESSIONE DI DATI E NELL'ACCESSO AD ESSI.

VEDREMO SOLO ALCUNE SEMPLICI APPLICAZIONI DELLA CLASSE TTREE, PER UN COMPLETO APPROFONDIMENTO SI VEDA ROOT USER GUIDE DISPONIBILE IN PDF SUL SITO DI ROOTCERN, CAP 12.

NELLA LEZIONE 2 ABBIAMO VISTO COME CREARE GRAFICI ESTRAENDO DATI DA FILE.TXT PER ESEGUIRE AD ESEMPIO UN FIT LINEARE.
VEDIAMO ORA COME SALVARE QUESTE INFORMAZIONI CIOÈ (DATI X ,DATI Y , ERRORE X ,ERRORE Y) IN UN UNICO FILE.ROOT.

Struttura Tree



- possiamo immagazzinare i dati all'interno di un Tree organizzandoli in vari Branches.
- Ciascun Branch può a sua volta contenere variabili diverse (int ,float,double….) organizzate in leaf.
- Convenzionalmente si tende a catalogare variabili indipendenti in Branches differenti.

- **Macros**
- **chiamata della clase TFile in cui creiamo un file chiamato: esTree.root nel path specifico**
- **classe TTree creazione dell ‘oggetto t1**

```
void  T(){
    cout<<"Root Cern tree a simple exercise "<<endl;
    //creazione di un tree
    TFile *ff =  new TFile("/Users/David/root/macros/esTree.root","RECREATE");
    //creo un file esTree.root nel path indicato
    TTree *t1 = new TTree ("t1","example tree");
    //creo un tree t1 dal titolo "example tree"
    struct dato {
        float  x;
        float errx;
    };

    dato dx;
    dato dy;

    t1->Branch("xx",&dx,"valore/F:errore");
    //2 leaf uguali conta l'ordine il primo valore che assume dx va su valore
    t1->Branch("yy",&dy,"valore/F:errore");

    fstream g;
```

- **Struct definisce un nuovo tipo di variabile costituita da 2 o più sottovariabili .Es: la struct dato possiede al suo interno la variabile reale x(valore sperimentale) e errx (errore sperimentale associato).**
- **Il Tree creato è costituito da 2 Branches contenenti 2 leafs ciascuno: “xx” nome del branch, &dx indirizzo dell’oggetto dx (struct) , “valore/F:errore” nomi dei 2 leaf , “/F” definisce le variabili: in questo caso float “/F”.**

```
    fstream g;
    g.open("/Users/David/Desktop/ROOT_AISF/tree/datay.txt",ios::in);
    fstream f;
    f.open("/Users/David/Desktop/ROOT_AISF/tree/datax.txt",ios::in);
    if (f.bad()) {
        cout<<"errore"<<endl;
    }
    while (!f.eof() && !g.eof() ) {
        f>>dx.x>>dx.errx>>ws;
        g>>dy.y>>dy.erry>>ws;
        t1->Fill();                }

    f.close(); g.close();

    t1->Print();
    ff->Write();
    ff->Close();
    cout<<"Root Cern tree created"<<endl;
    //chiusura automatica del file root
}
```

- importiamo dati da file txt salvandoli sulle variabili dx.x, dx.errx
- metodo Fill immagazzina i dati all'interno del Tree nei relativi Branch e leaf.
- t1->Print() salva il tree "completo" sul file esTree.root
- Eseguita la macros il file esTree.root contiene tutte le informazioni precedentemente salvate nei vari file.txt

- per condividere i dati relativi ad un esperimento basta dunque condividere il solo file.root contenente il Tree sul quale abbiamo inserito nei vari branches tutti i dati utili.
- Vediamo ora come estrarre i dati dal file.Root

- **salvato il file.root nel path specifico (“/root/macros/ ”) vediamo come visualizzare il Tree.**

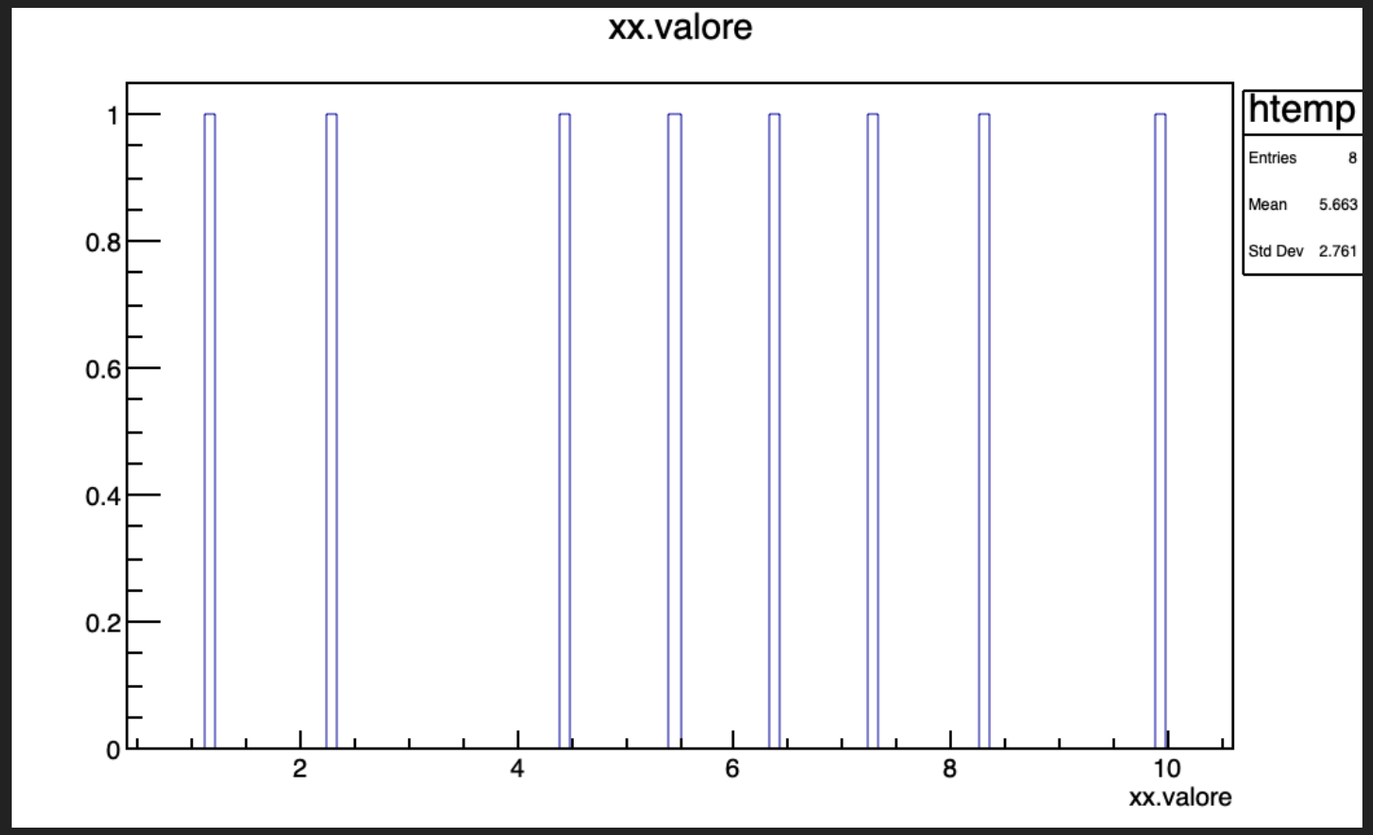
```
MacBook-Pro-di-David:macros David$
MacBook-Pro-di-David:macros David$ root
   ------------------------------------------------------
  | Welcome to ROOT 6.12/06                http://root.cern.ch |
  |                                (c) 1995-2017, The ROOT Team |
  | Built for macosx64                                          |
  | From tag v6-12-06, 9 February 2018                          |
  | Try '.help', '.demo', '.license', '.credits', '.quit'/'.q' |
   ------------------------------------------------------

root [0] .L TreedataFit.cpp
root [1] T
Root Cern tree a simple exercise
*******************************************************************************
*Tree    :t1        : example tree                                           *
*Entries :       16 : Total =            1956 bytes  File  Size =          0 *
*        :          : Tree compression factor =   1.00                       *
*******************************************************************************
*Br    0 :xx        : x/F:errx                                               *
*Entries :       16 : Total  Size=        825 bytes  One basket in memory    *
*Baskets :        0 : Basket Size=      32000 bytes  Compression=   1.00     *
*.............................................................................*
*Br    1 :yy        : y/F:erry                                               *
*Entries :       16 : Total  Size=        825 bytes  One basket in memory    *
*Baskets :        0 : Basket Size=      32000 bytes  Compression=   1.00     *
*.............................................................................*
Root Cern tree created
root [2] TFile w("esTree.root")
(TFile &) Name: esTree.root Title:
root [3] TBrowser a
(TBrowser &) Name: Browser Title: ROOT Object Browser
root [4]
```

- **aprire la macros ed eseguirla**
- **aprire il file.root tramite la classe TFile**
- **aprire la finestra TBrowser**

- dalla finestra Browser selezionare StartViewer

- otteniamo il seguente risultato : visualizziamo i branches (rami)xx e yy e i vari leaf (foglie).



- **selezionando ad esempio xx.valore vediamo tutte le variabili “storate” all’interno del Tree contenute nel leaf xx.valore**

- **Vengono visualizzate con istogramma.**

- **Come estrarre i dati dal Tree?**

```
void  T(){
    struct dato {
        float  x;
        float errx;
    };

    dato dx,dy;

    auto infile = TFile::Open("/Users/David/root/macros/esTree.root");

    auto tree = (TTree *)infile->Get("t1");

    tree->SetBranchAddress("xx",&dx);
    tree->SetBranchAddress("yy",&dy);
```

Macros per importazioni dati da un Tree

- open il file.root che contiene il Tree
- creiamo un tree “interno” alla macros estraendolo (Get(“t1”)) dal file root. Definiamo le struct dx e dy
- Metodo SetBranchAddress per “associare” il branch del tree agli oggetti (struct)dx e dy.

```
    int nEv = tree->GetEntries();
    for (int iEv= 0 ; iEv < nEv; iEv++){
        tree->GetEntry(iEv);
        cout<<"---> "<<dy.x<<"    "<<dy.errx<<endl;
    }

    return 0;
}
```

- **tree->GetEntries()**
- **ad ogni iterazione dy.x conterrà il valore i-esimo contenuto nel leaf valore del branch yy.**
- **ad ogni iterazione dy.errx conterrà il valore i-esimo contenuto nel leaf errore del branch yy.**

```
---> 1.2    0.2
---> 2.3    0.2
---> 4.4    0.2
---> 5.5    0.2
---> 6.4    0.2
---> 7.3    0.2
---> 8.3    0.2
---> 9.4    0.2
```

- **riottenendo i valori contenuti inizialmente nel file data.txt estratti da un file.root**

- **Nella Macros createTree.cpp esempio su come creare il file.root contenente il tree a partire da un generico file.txt**
- **Nella Macros FitfromTree.cpp esempio su come eseguire un grafico a partire da dati salvati in file.root**