PROOF tutorial
EventTree

Gerardo Ganis, CERN, PH-SFT
gerardo.ganis@cern.ch

Introduction

- To show how to work with trees we will play with a tree of the class Event defined by

  eventclass/Event.h
  eventclass/Event.cxx

  which has the structure of a possible HEP event:
  - An header (→ class EventHeader)
  - Array of tracks (→ class Track)
  - Some control information
- It allows also to see the effect of splitting
- Let's dissect the macro CreateEventTree.C under macros

CreateEventTree.C (1)

- Purpose:
  - Create a configurable number of files with a TTree named 'EventTree' whose unique branch has a class Event
    - Filename: event_tree_j.root (event_tree_j_unsplit.root)
  - Create a friend 'EventFriend' for each of the files
    - Filename: event_friend_j.root
- Allow to choose
  - split/unsplit
  - location of files
  - number of entries

CreateEventTree.C (2)

- Signature

```
void CreateEventTree(const char *basedir,
                     Int_t nevents,
                     Int_t nfiles,
                     Int_t first = 1,
                     Bool_t split = kTRUE)
```

  - Files under basedir
  - nevents per file
  - Generate nfiles
  - First file ordinal is first
  - Split if split is kTRUE

CreateEventTree.C (3)

- Filename generation and initialization of the random generator

```
// Seed
TString seed;
seed.Form("%s_%d", ord.Data(), i);
if (!split) seed += "_unsplit";
gRandom->SetSeed(static_cast<UInt_t>(TMath::Hash(seed)));

// File names
TString filename, filefriend;
filename.Form("%s/event_tree_%s.root", basedir, seed.Data());
filefriend.Form("%s/event_friend_%s.root", basedir, seed.Data());
```

  - Each file gets a different seed

CreateEventTree.C (4)

- Create the files, deleting existing files, if any ...

```
TFile *f = TFile::Open(filename, "RECREATE");
TFile *ff = TFile::Open(filefriend, "RECREATE");
```

- … and the trees, attaching them to the related file for automatic swapping

```
TTree eventtree("EventTree", "Event Tree");
eventtree.SetDirectory(f);
eventtree.AutoSave();

TTree eventfriend("EventFriend", "Event Friend");
eventfriend.SetDirectory(ff);
eventfriend.AutoSave();
```

CreateEventTree.C (5)

- Create the branch for the main tree; splitted, if required

```
Event event;
Event *ep = &event;
if (split) {
   eventtree.Branch("event", "Event", &ep);
} else {
   eventtree.Branch("event", "Event", &ep, 32000, 0);
}
```

- Create the branch for the friend tree

```
Double_t temp;
eventfriend.Branch("temp", &temp, "temp/D");
```

CreateEventTree.C (6)

- Generate the events and fill the trees

```
for (Int_t j = 0; j < nevents; j++) {
   event.Build(j,50,1);
   temp = event.GetTemperature();
   eventtree.Fill();
   eventfriend.Fill();
}
```

- Build generates randomly the event
  - (No physics in it …)
- Memory is automatically flushed to file during filling, if needed

CreateEventTree.C (7)

- Final flush (writing) and file closure

```
// Main tree
f->cd();
eventtree.Write(0, TObject::kOverwrite);
eventtree.SetDirectory(0);
f->Close();
delete f;
f = 0;

// Friend tree
ff->cd();
eventfriend.Write(0, TObject::kOverwrite);
eventfriend.SetDirectory(0);
ff->Close();
delete ff;
ff = 0;
```

  - We make sure that there is only one copy of the tree (kOverwrite)

Try it out

- Needs to load Event first ...

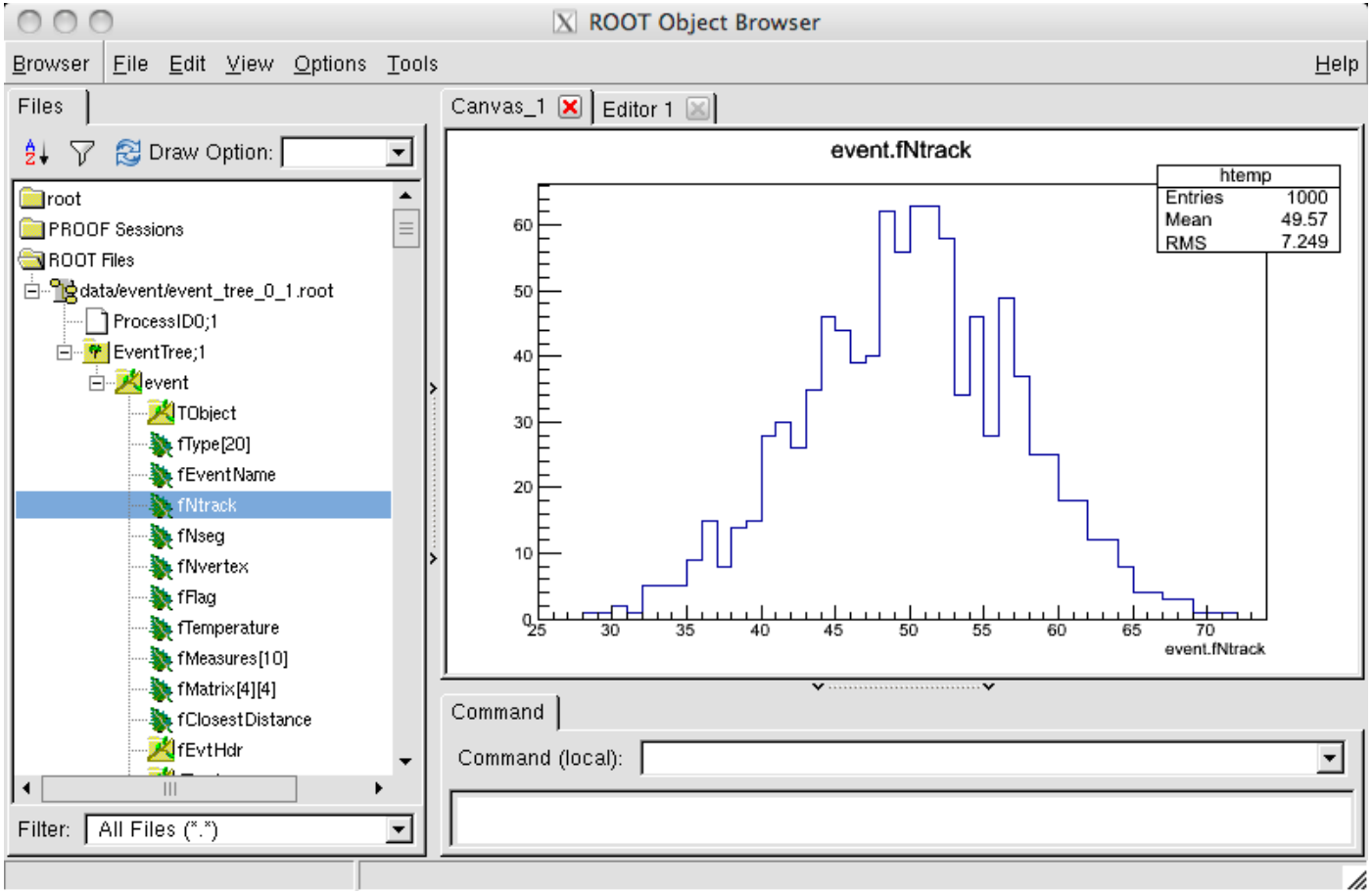
```
root [1] .L eventclass/Event.cxx+
Info in <TUnixSystem::ACLiC>: creating shared library ...
root [2] .L macros/CreateEventTree.C+
Info in <TUnixSystem::ACLiC>: creating shared library …
```

- … then we can run

```
root [3] CreateEventTree("data/event", 10000, 10)
CreateEventTree: opening file data/event/event_tree_0_1.root ...
0x1047446f0 data/event/event_friend_0_1.root
CreateEventTree: opening file data/event/event_tree_0_2.root ...
0x104940e50 data/event/event_friend_0_2.root
CreateEventTree: opening file data/event/event_tree_0_3.root ...
0x104954100 data/event/event_friend_0_3.root
```

Check the file with the browser

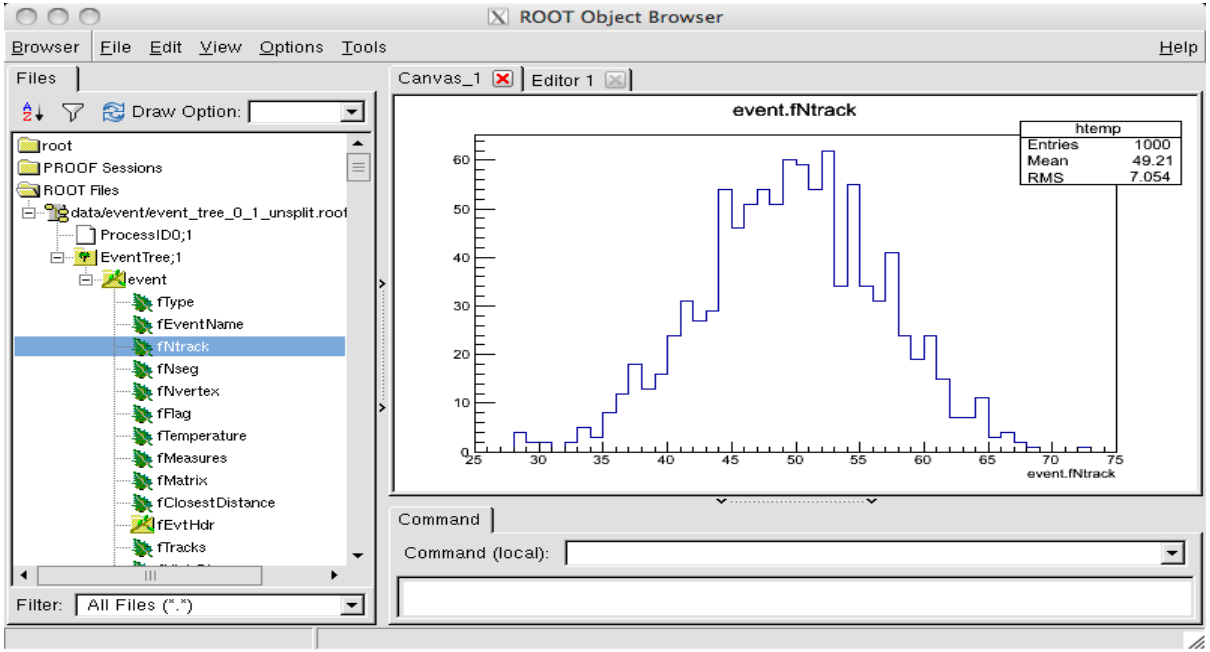
```
root [4] f = TFile::Open("data/event/event_tree_0_1.root")
(class TFile*)0x104923260
root [5] TBrowser b
```



Rerun in unsplit mode

```
root [3] CreateEventTree("data/event", 10000, 10, 1, kFALSE)
CreateEventTree: opening file data/event/event_tree_0_1_unsplit.root ...
0x104fdf680 data/event/event_friend_0_1_unsplit.root
CreateEventTree: opening file data/event/event_tree_0_2_unsplit.root ...
0x104fdf620 data/event/event_friend_0_2_unsplit.root
CreateEventTree: opening file data/event/event_tree_0_3_unsplit.root ...
0x104fdf620 data/event/event_friend_0_3_unsplit.root
...
```

- The browser is still able to see the leaves ...



Split vs Unsplit

- But if you check with TTree::Print() you can see the difference

```
root [0] f = TFile::Open("data/event/event_tree_0_1_unsplit.root")
(class TFile*)0x101694cf0
root [1] EventTree.Print()
******************************************************************************
*Tree    :EventTree : Event Tree                                             *
*Entries :     1000 : Total =         7541209 bytes  File  Size =    3391707 *
*        :          : Tree compression factor =   2.22                       *
******************************************************************************
*Br    0 :event     : Event                                                   *
*Entries :     1000 : Total  Size=    7540808 bytes  File Size  =    3388990 *
*Baskets :      260 : Basket Size=      32000 bytes  Compression=   2.22     *
*.............................................................................*
```

Split vs Unsplit (2)

```
root [0] f = TFile::Open("data/event/event_tree_0_1.root")
(class TFile*)0x101695150
root [1] EventTree.Print()
******************************************************************************
*Tree    :EventTree : Event Tree                                             *
*Entries :     1000 : Total =         7238869 bytes  File  Size =    2679857 *
*        :          : Tree compression factor =   2.70                       *
******************************************************************************
*Branch  :event                                                              *
*Entries :     1000 : BranchElement (see below)                              *
*.............................................................................*
*Br    0 :fUniqueID : UInt_t                                                 *
*Entries :     1000 : Total  Size=       4575 bytes  File Size  =        131 *
*Baskets :        1 : Basket Size=      32000 bytes  Compression=  31.15     *
*.............................................................................*
*Br    1 :fBits     : UInt_t                                                 *
*Entries :     1000 : Total  Size=       8563 bytes  File Size  =       1391 *
*Baskets :        1 : Basket Size=      32000 bytes  Compression=   5.81     *
*.............................................................................*
...
*.............................................................................*
*Br   54 :fTriggerBits.fAllBits : UChar_t fTriggerBits.fAllBits[fNbytes]     *
*Entries :     1000 : Total  Size=      17104 bytes  File Size  =       7062 *
*Baskets :        1 : Basket Size=      32000 bytes  Compression=   2.33     *
*.............................................................................*
*Br   55 :fIsValid  : Bool_t                                                 *
*Entries :     1000 : Total  Size=       1570 bytes  File Size  =        108 *
*Baskets :        1 : Basket Size=      32000 bytes  Compression=  10.00     *
*.............................................................................*
```