ROOT I/O Overview

CMS-ROOT meeting

CERN- October 10

René Brun

ftp://root.cern.ch/root/cms.ppt

http://root.cern.ch

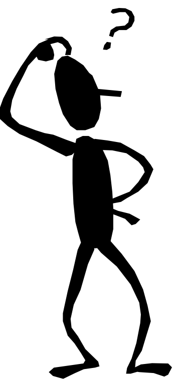
http://root.cern.ch/root/RootDoc.html

Plan of talk

- Framework Structure
- The Object Dictionary and the rootcint tool
- Persistency basics:
  - Evolution of ROOT I/O
  - Basic Object I/O
  - CMS PSimHit example
  - File Structure
  - Streamers
  - Automatic Schema Evolution
- Special collection Classes
- Trees
  - Event example
  - PSimHit example
  - Tree Friends
  - Chains
- Folders

Motivations

Towards the ROOT Framework



Following our many years of experience with the development of the PAW system, we decided in 1995 to start the design and the implementation of a system capable of doing at least the same thing in an OO context, but also to serve as a complete framework from data taking to data analysis.

During a few months, we learnt the basics ingredients of an OO system by implementing several variants of an histograming package. We quickly implemented a rudimentary I/O sub system and also some very basic collection classes. It became rapidly clear to us that a more ambitious persistency mechanism had to be developed.

There was no point in developing a system supporting only the PAW CWNs in a world dealing with classes and complex object hierarchies. OODBMS could have been the solution to our problem, but we were convinced that the corresponding proposed commercial tools were not appropriate for a flexible data analysis environment.

Building a Modular System

Modularity is a buzzword with different meanings.. A modular system is sometimes presented as a system with many small and independent components. In general such systems do not have an object bus and the communication between the components is left to the application using these components.

Systems with a deep hierarchy of components may be difficult to maintain because of too many interdependencies between the top level and low level modules.

Is a system with well defined interfaces a modular system? Probably not, because too much emphasis is put on the interfaces at the expense of the object bus. In such systems, the interfaces may have long argument lists instead of well designed collections and object folders.

An end user will see a system as modular if the structure is easy to understand, while a system developer will put more emphasis on the maintenance aspects, probably the two aspects being strongly related. A modular system can also be seen as a system easy to integrate into another system.

After many iterations and user feedback, we have gradually converged to the following framework structure ==>

ROOT Framework Organization

$ROOTSYS

bin

cint
makecint
new
proofd
proofserv
rmkdepend
root
root.exe
rootcint
root-config
rootd

* Optional Installation

lib

libCint.so
libCore.so
libEG.so
*libEGPythia.so
*libEGPythia6.so
libEGVenus.so
libGpad.so
libGraf.so
libGraf3d.so
libGui.so
libGX11.so
*libGX11TTF.so
libHist.so
libHistPainter.so
libHtml.so
libMatrix.so
libMinuit.so
libNew.so
libPhysics.so
libPostscript.so
libProof.so
*libRFIO.so
*libRGL.so
libRint.so
*libThread.so
libTree.so
libTreePlayer.so
libTreeViewer.so
*libttf.so
libX3d.so
libXpm.a

tutorials

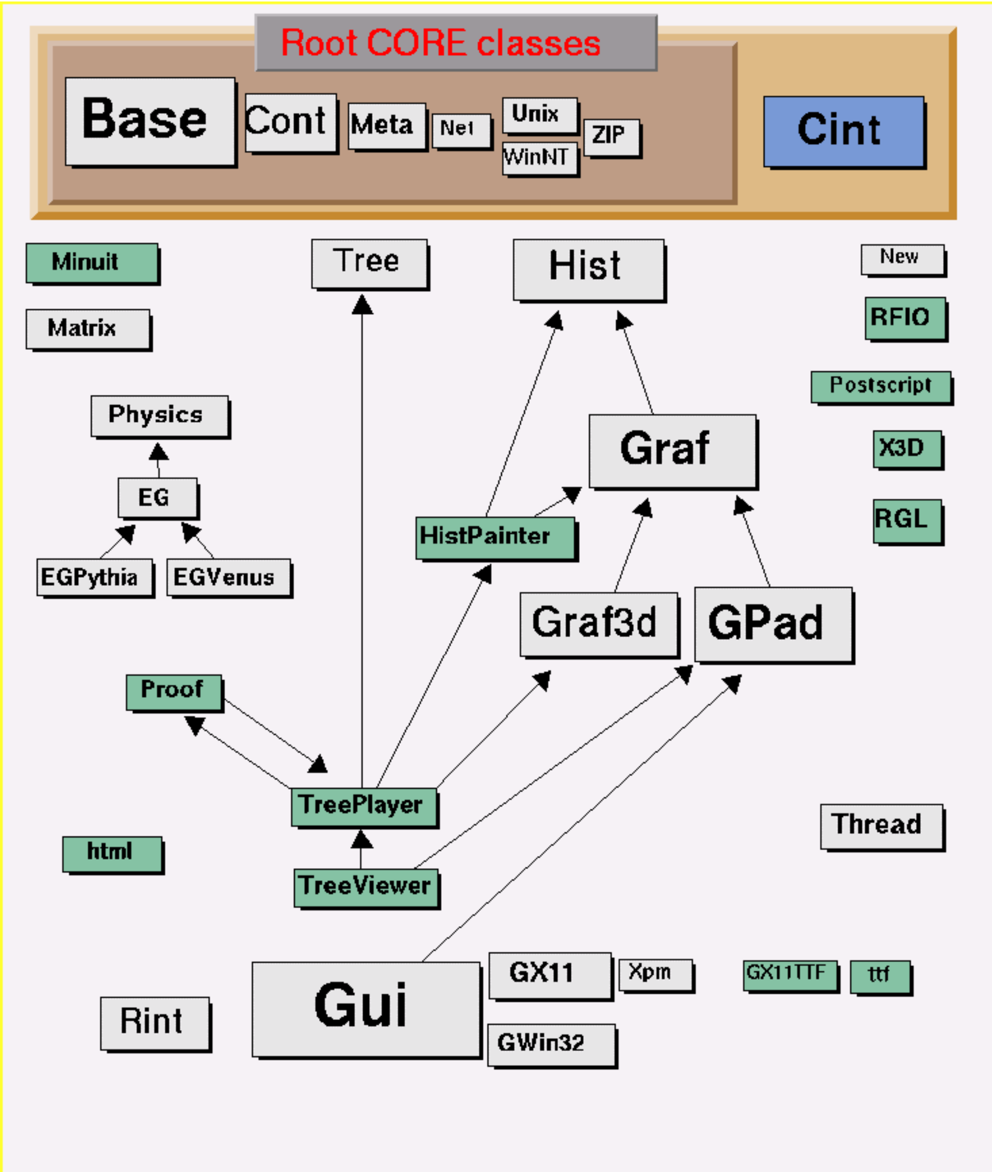
EditorBar.C
Ifit.C
analyze.C
archi.C
arrow.C
basic.C
basic.dat
basic3d.C
benchmarks.C
canvas.C
classcat.C
cleanup.C
compile.C
copytree.C
copytree2.C
demos.C
demoshelp.C
dialogs.C
dirs.C
ellipse.C
eval.C
event.C
exec1.C
exec2.C
feynman.C
fildir.C
file.C
fillrandom.C
first.C
fit1.C
fit1_C.C
fitslicesy.C
formula1.C
framework.C
games.C
gaxis.C
geometry.C
gerrors.C
gerrors2.C
graph.C
h1draw.C
hadd.C
hclient.C
hcons.C
hprod.C
hserv.C
hserv2.C
hsimple.C
hsum.C
hsumTimer.C
htmlex.C
io.C
latex.C
latex2.C
latex3.C
manyaxis.C
multifit.C
myfit.C
na49.C
na49geomfile.C
na49view.C
na49visible.C
ntuple1.C
oldbenchmarks.C
pdg.dat
psexam.C
pstable.C
rootalias.C
rootenv.C
rootlogoff.C
rootlogon.C
rootmarks.C
runcatalog.sql
runzdemo.C
second.C
shapes.C
shared.C
splines.C
sqlcreatedb.C
sqlfilldb.C
sqlselect.C
staff.C
staff.dat
surfaces.C
tcl.C
testrandom.C
tornado.C
tree.C
two.C
xyslider.C
xysliderAction.C
zdemo.C

test

Aclock.cxx
Aclock.h
Event.cxx
Event.h
EventLinkDef.h
Hello.cxx
Hello.h
MainEvent.cxx
Makefile
Makefile.in
Makefile.win32
README
TestVectors.cxx
Tetris.cxx
Tetris.h
eventa.cxx
eventb.cxx
eventload.cxx
guitest.cxx
hsimple.cxx
hworld.cxx
minexam.cxx
stress.cxx
tcollbm.cxx
tcollex.cxx
test2html.cxx
tstring.cxx
vlazy.cxx
vmatrix.cxx
vvector.cxx

include

*.h
...



The Libraries

- Over 500 classes

- 650,000 lines of code
- Core (5 Mbytes)
- CINT (1.5 Mbytes)
- All libs (17 Mbytes)
- green libs linked on demand

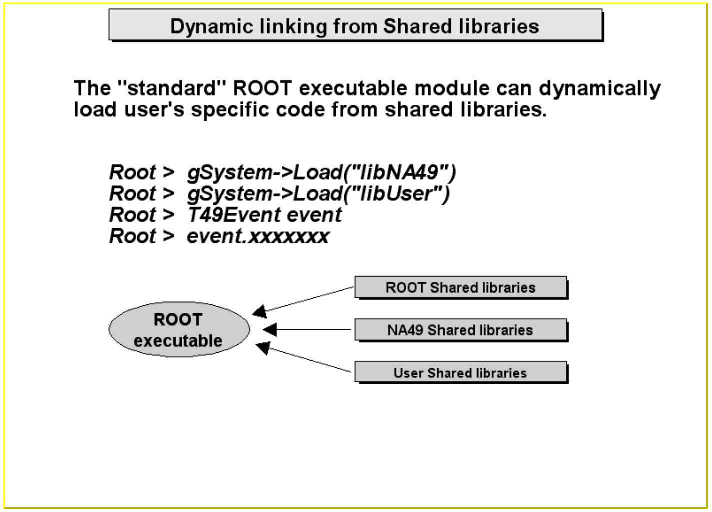
Root Libs Structure

- Root libs are a layered structure
- CORE classes always required (support for RTTI, basic I/O and interpreter.
- The application libraries. <u>You load only what you use</u>. Separation between Data Objects and the high level classes acting on these objects. Example, a batch job uses only the Hist lib, no need to link HistPainter.
- Root shared libs reduce the application link time.
- Root libs are small libraries.
- Root libs can be used with other class libraries.

Dynamic Linking

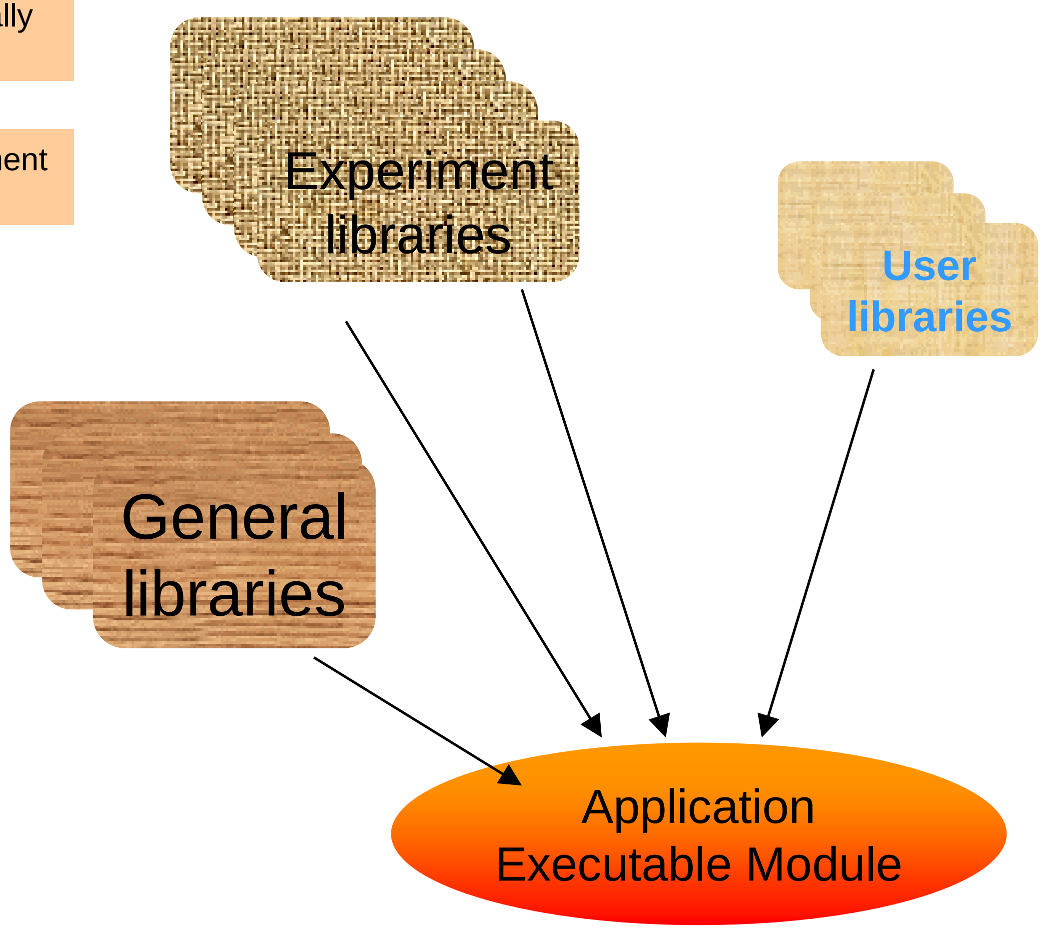
A Shared Library can be linked dynamically to a running executable module

A Shared Library facilitates the development and maintenance phases


Dynamic linking from Shared libraries

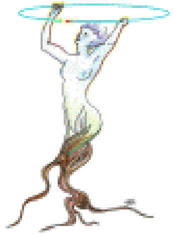
The "standard" ROOT executable module can dynamically load user's specific code from shared libraries.

```
Root >  gSystem->Load("libNA49")
Root >  gSystem->Load("libUser")
Root >  T49Event event
Root >  event.xxxxxxx
```

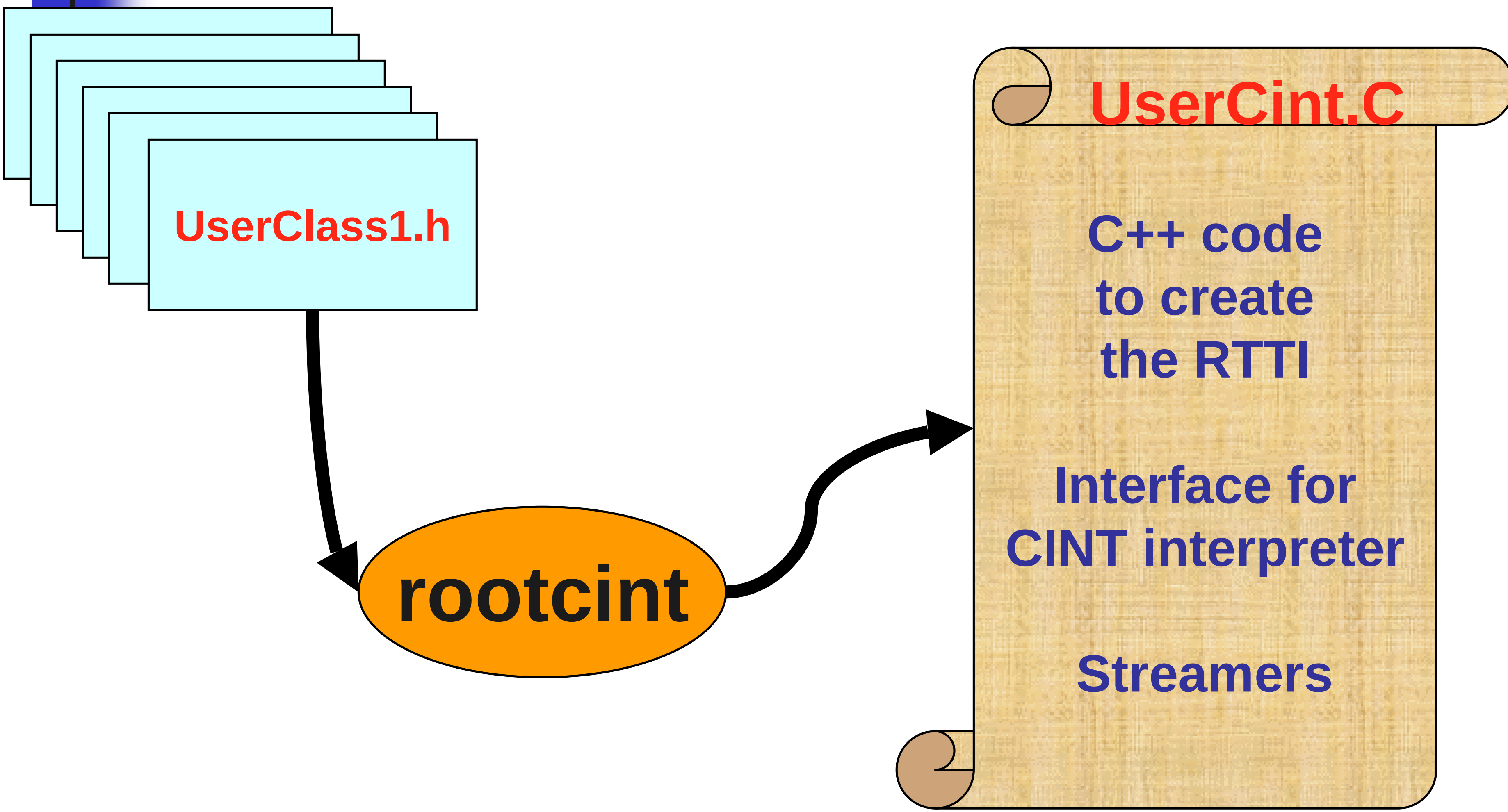
The ROOT Object Dictionary

The CINT RTTI

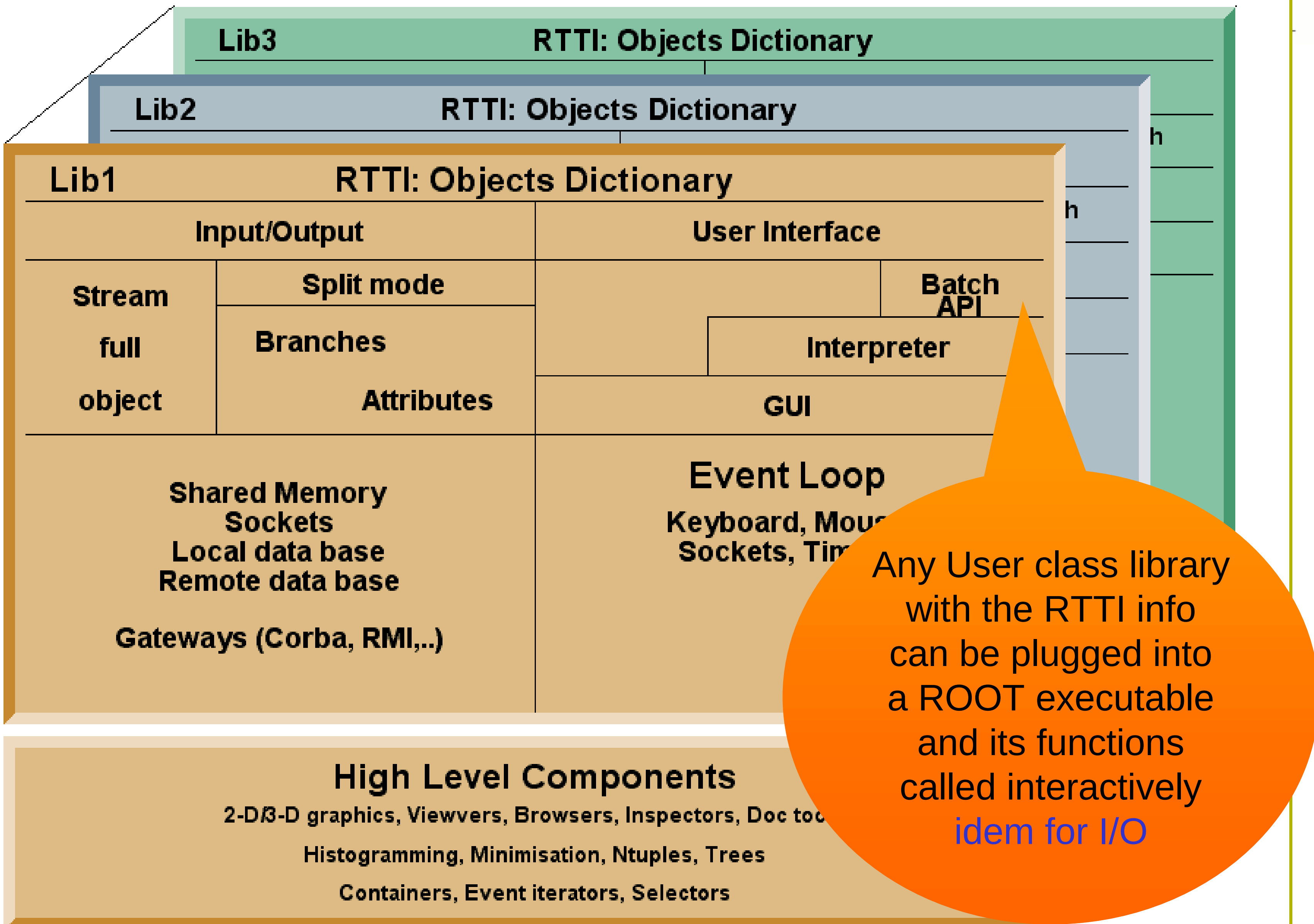


- The Run Time Type Information provided by CINT (rootcint) is the brain of Root. rootcint can be used to parse user classes and considerably extend the power of Root.
- RTTI is used by the I/O services
- By definition the interpreter is based on it.
- The GUI object context sensitive menus also.
- Also Browsers, Inspectors and html generator
- also Root utilities to draw class diagrams
- rootcint can be used to parse user classes such that user class functions can be called interactively and code for I/O generated automatically.

Rootcint Preprocessor

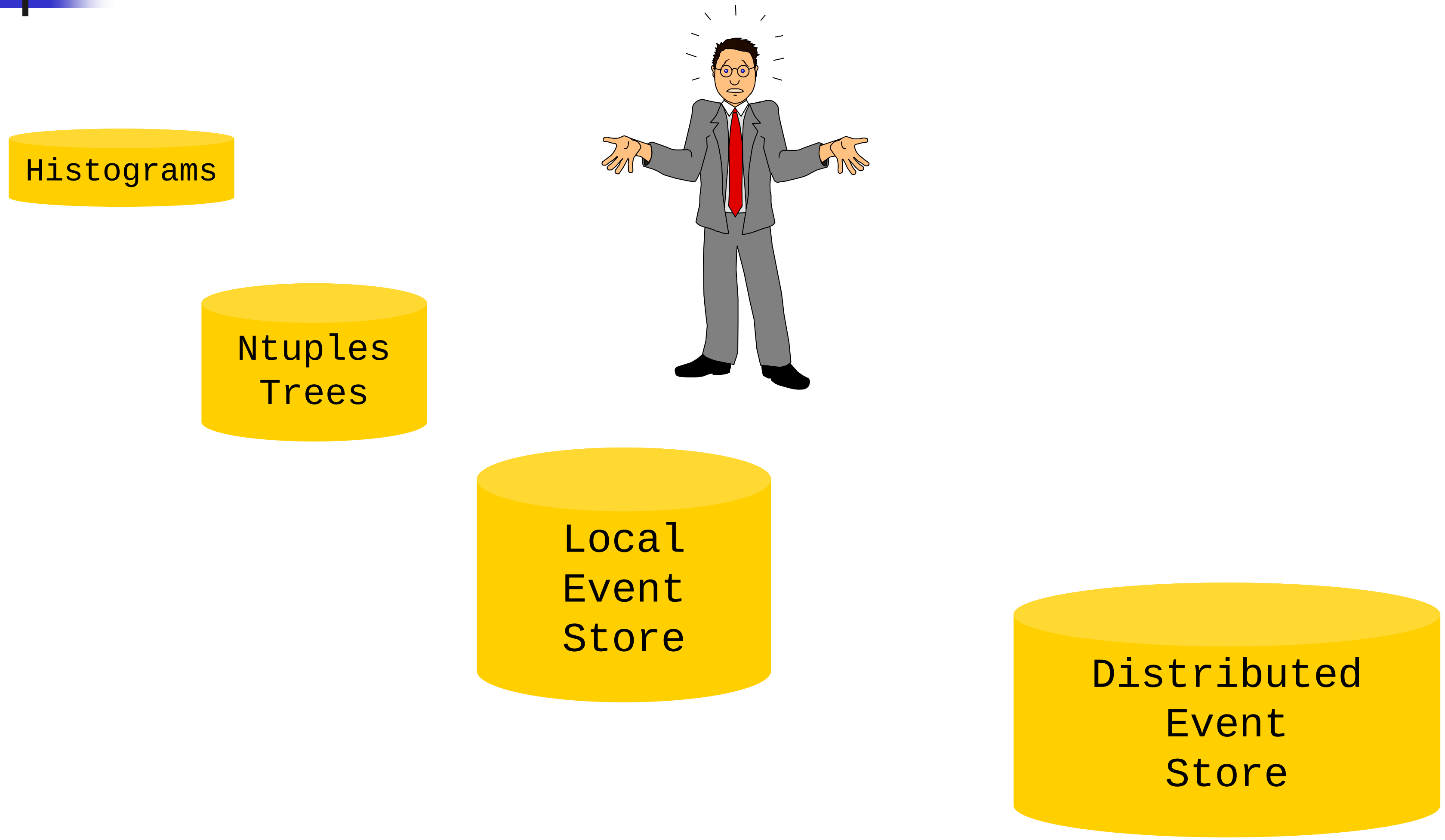


Framework: Basic components

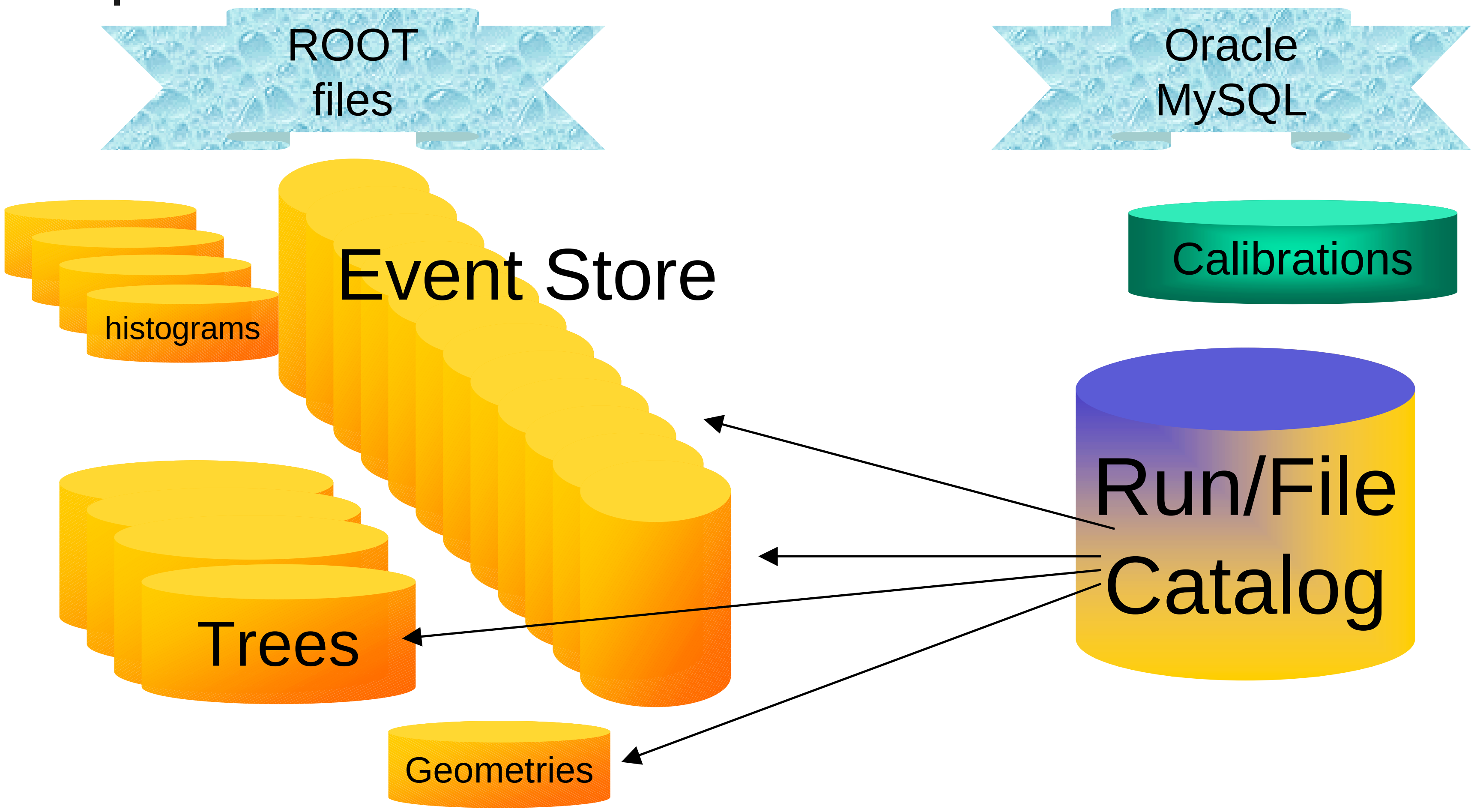


Object Persistency

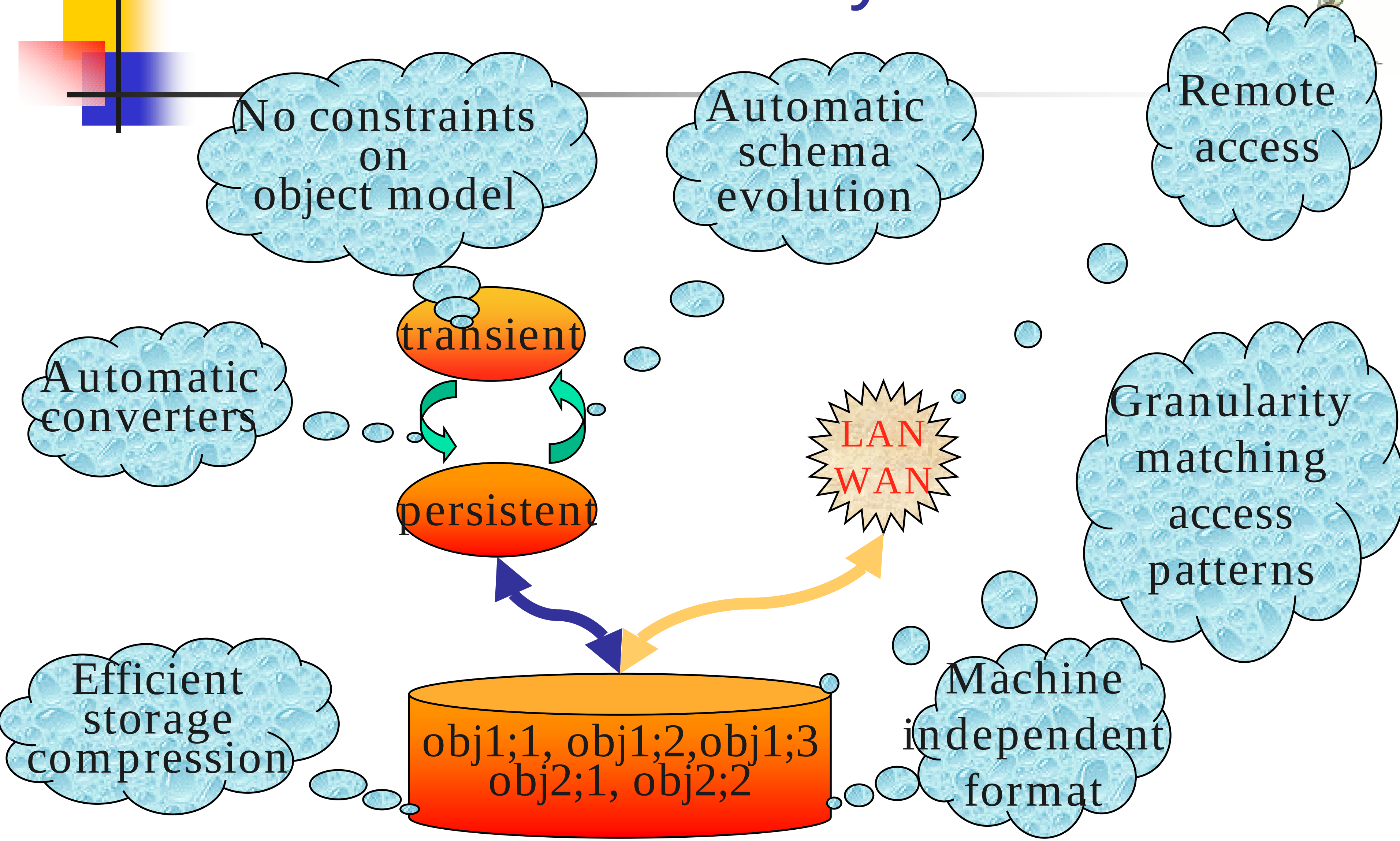
Simple to Complex cases



ROOT + RDBMS Model



Ideal Persistency



Evolution of ROOT I/O

1995

- Hand-written Streamers
- Streamers generated via rootcint
- Support for Class Versions
- Support for ByteCount
- Several attempts to introduce automatic class evolution
- Persistent class Dictionary written to files
- rootcint modified to generate automatic Streamers
- can generate code for “DataObjects” classes in a file
- Support for STL and more complex C++ cases
- Trees take advantage of the new scheme
- Can read files without the classes
- Persistent Reference pointers New

3.00

3.01

3.02

2001

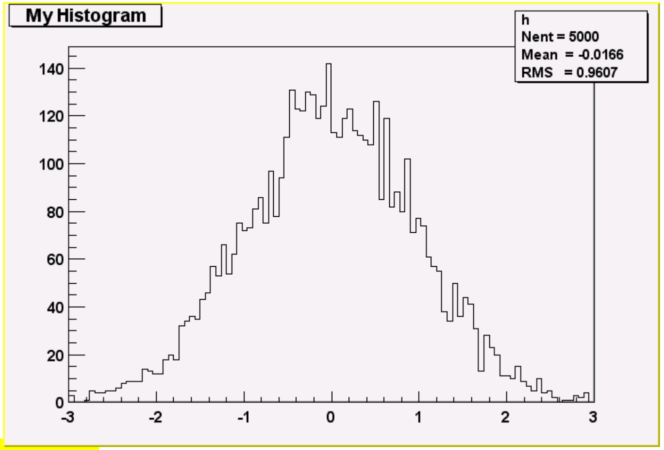
ROOT I/O : An Example

Program Writing

```
TFile f("example.root","new");
TH1F h("h","My histogram",100,-3,3);
h.FillRandom("gaus",5000);
h.Write();
```

Program Reading

```
TFile f("example.root");
TH1F *h = (TH1F*)f.Get("h"):
h->Draw();
f.Map();
```




```
20010831/171903  At:64     N=90     TFile
20010831/171941  At:154    N=453    TH1F          CX =  2.09
20010831/171946  At:607    N=2364   StreamerInfo  CX =  3.25
20010831/171946  At:2971   N=96     KeysList
20010831/171946  At:3067   N=56     FreeSegments
20010831/171946  At:3123   N=1      END
```

Example with CMS classes

- In the following slides we will use the CMS simulation classes PSimHit, etc.
- Classes suggested by Vincenzo as exercise.
- Goal: No changes in the class model
  - Minor changes in the header files to make these classes ROOT-aware
- Example1: How to generate the dictionary
- Example2: How to write PSimHit objects to a ROOT file (and read)
- Example3: How to write PSimHit objects to a Tree (and read)

ftp://root.cern.ch/root/cmsdemo.tar.gz

PSimHit.h

```
#include "LocalPoint.h"
#include "LocalVector.h"
#include "TObject.h"

class DetUnit;

class PSimHit : public TObject {
public:

  PSimHit() : theDetUnitId(-1) {}
  PSimHit( const Local3DPoint& entry, const Local3DPoint& exit,
                float pabs, float tof, float eloss, int particleType,
                int detId, unsigned int trackId) :
  theEntryPoint( entry), theExitPoint(exit),
.....

  float        pabs()           const {return thePabs;}
  float        tof()            const {return theTof;}
  float        energyLoss()     const {return theEnergyLoss;}
  int          particleType()   const {return theParticleType;}
  int          detUnitId()      const {return theDetUnitId;}
  unsigned int trackId()        const {return theTrackId;}

private:

  // properties
  Local3DPoint    theEntryPoint;    // position
  Local3DPoint    theExitPoint;
  float           thePabs;          // momentum
  float           theTof;           // Time Of Flight
  float           theEnergyLoss;    // Energy loss
  int             theParticleType;

  // association
  int             theDetUnitId;
  unsigned int    theTrackId;

ClassDef(PSimHit,1)
};
```

```
#include "LocalTag.h"
#include "Point2DBase.h"
#include "Point3DBase.h"

typedef Point2DBase< float, LocalTag>    Local2DPoint;
typedef Point3DBase< float, LocalTag>    Local3DPoint;

// Local points are two-dimensional by default
typedef Local3DPoint                LocalPoint;
```

Point3DBase.h

```
#include "PV3DBase.h"
#include "Point2DBase.h"
#include "Vector3DBase.h"
#include "TObject.h"

template <class T, class FrameTag>
class Point3DBase : public PV3DBase< T, PointTag, FrameTag> {
public:

  typedef PV3DBase< T, PointTag, FrameTag>    BaseClass;
  typedef Vector3DBase< T, FrameTag>          VectorType;
  typedef Basic3DVector<T>                    BasicVectorType;

  Point3DBase() {}

  Point3DBase(const T& x, const T& y, const T& z) : BaseClass(x, y, z)
{}

…….
ClassDefT(Point3DBase,1)
};
ClassDef2T2(Point3DBase,T,FrameTag)
```

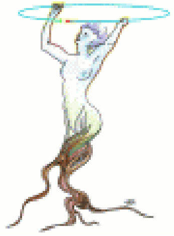
Point3DBase.h

```
#include "Basic3DVector.h"
#include <iosfwd>
#include "TObject.h"

template <class T, class PVType, class FrameType>
class PV3DBase {
public:
  typedef Basic3DVector<T> BasicVectorType;

  PV3DBase() : theVector() {}
  PV3DBase(const T & x, const T & y, const T & z) : theVector(x, y, z) {}
  PV3DBase( const Basic3DVector<T>& v) : theVector(v) {}

  T x()    const { return basicVector().x();}
  T y()    const { return basicVector().y();}
  T mag2() const { return basicVector().mag2();}
  T r()    const { return basicVector().r();}
…….
protected:
  BasicVectorType& basicVector() { return theVector;}

private:
  BasicVectorType theVector;

ClassDefT(PV3DBase,1)
};

ClassDef3T2(PV3DBase,T,PVType,FrameType)
```

Basic3DVector.h

```
#include "Basic2DVector.h"
#include <iosfwd>
#include <cmath>
#include "TObject.h"

template < class T>
class Basic3DVector {

public:
  // default constructor
  Basic3DVector() : theX(0), theY(0), theZ(0){}

  Basic3DVector( const T& x, const T& y, const T& z) :
    theX(x), theY(y), theZ(z) {}

  T x() const { return theX;}
  T y() const { return theY;}
  T z() const { return theZ;}
…...
private:
  T theX;
  T theY;
  T theZ;

ClassDefT(Basic3DVector,1)
};
ClassDefT2(Basic3DVector,T)
```

Running rootcint on PSimHit.h
Building the shared lib



```
rootcint -f Dict.cxx -c PSimHit.h LinkDef.h
g++ -fPIC -I$ROOTSYS/include -c Dict.cxx
g++ -fPIC -I$ROOTSYS/include -c PSimHit.cxx
g++ -shared -g PSimHit.o Dict.o -o libHit.so
```

LinkDef.h

```
#pragma link off all globals;
#pragma link off all classes;
#pragma link off all functions;

#pragma link C++ class LocalTag+;
#pragma link C++ class PointTag+;
#pragma link C++ class VectorTag+;
#pragma link C++ class Basic2DVector<float>+;
#pragma link C++ class Basic3DVector<float>+;
#pragma link C++ class PV2DBase<float, VectorTag, LocalTag>+;
#pragma link C++ class PV2DBase<float, PointTag, LocalTag>+;
#pragma link C++ class PV3DBase<float, VectorTag, LocalTag>+;
#pragma link C++ class PV3DBase<float, PointTag, LocalTag>+;
#pragma link C++ class Vector2DBase<float, LocalTag>+;
#pragma link C++ class Vector3DBase<float, LocalTag>+;
#pragma link C++ class Point2DBase<float, LocalTag>+;
#pragma link C++ class Point3DBase<float, LocalTag>+;
#pragma link C++ class PSimHit+;
```

Classes used by PSimHit
use C++ templates
heavily
All Template instances
must be declared

Writing CMS PSimHit objects

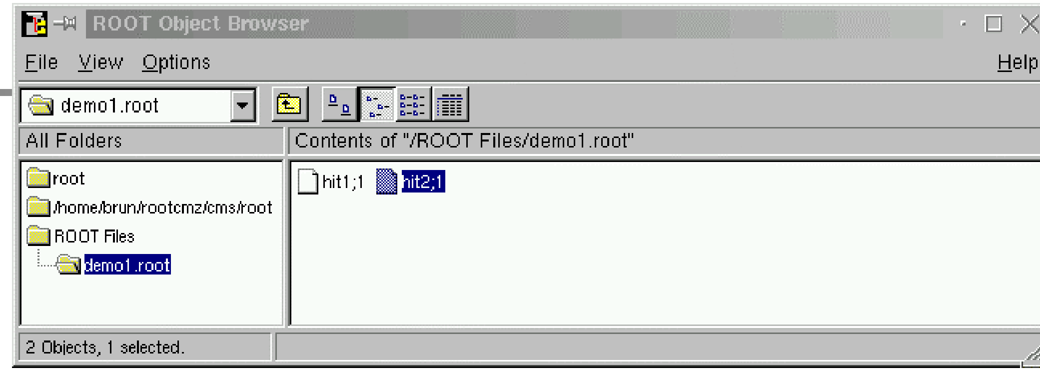
```
void demo1() {
   //create a new ROOT file
   TFile f("demo1.root","recreate");

   //Create a PSimHit with the default constructor
   PSimHit h1;

   //Write it to the file with the key name hit1
   h1.Write("hit1");

   //Create a normal PSimHit with the entry and exit point
   Local3DPoint pentry(1,2,3);
   Local3DPoint pexit(10,20,30);
   float pabs    = 41;
   float tof     = 1.67e-8;
   float eloss   = 5.78e-3;
   int   pType   = 12;
   int   detId   = 67;
   int   trackId = 1234;
   PSimHit h2(pentry,pexit,pabs,tof,eloss,pType,detId,trackId);

   //Write it to the file with the key name hit2
   h2.Write("hit2");

}
```

Reading CMS PSimHit objects

```
void demo2() {
   //connect the ROOT file demo1.root in readonly mode
   TFile *f = new TFile("demo1.root");

   //Read hit2
   PSimHit *hit = (PSimHit*)f->Get("hit2");

   //print some hit members
   cout <<" X1= "<<hit->entryPoint().x()
        <<" Y2= "<<hit->exitPoint().y()
        <<" pabs= "<<hit.pabs()<<endl;
   delete hit;

   //Open the ROOT browser and inspect the file
   new TBrowser;

   //click on "ROOT files", then "demo1.root", with the
  //right button, select menu items "Inspect", "DrawClass"
  //on hit2
}
```

Browsing the file

```
root [0] TFile f("demo1.root")
root [1] TBrowser b
root [2] f.ls();
```




```
TFile*          demo1.root
  KEY: PSimHit  hit1;1
  KEY: PSimHit  hit2;1
```

```
root [3] f.Map();
```

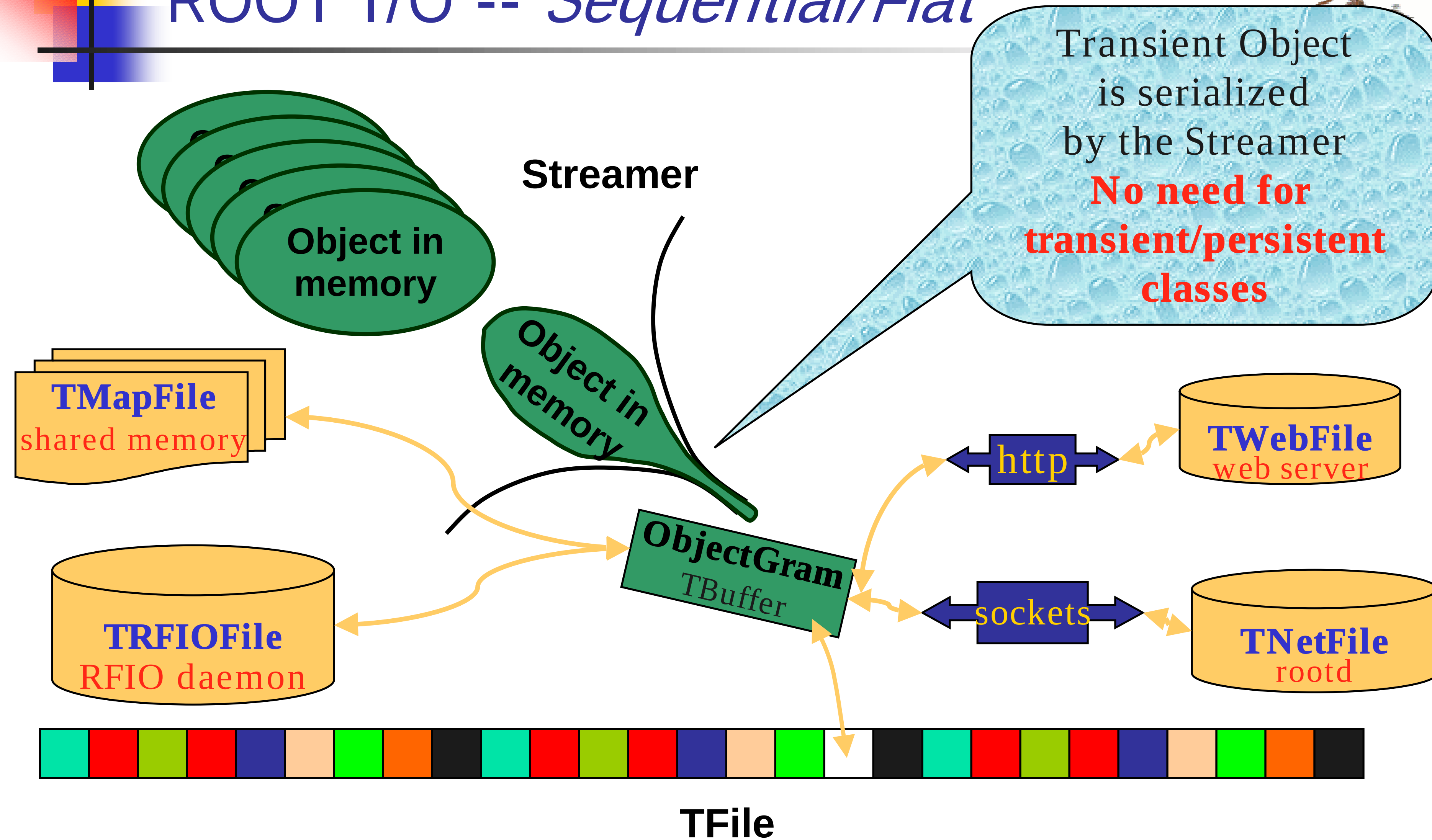
```
20011008/091050  At:64        N=86       TFile
20011008/091050  At:150       N=128      KeysList
Address = 278   Nbytes = -27    =====G A P===========
20011008/091050  At:305       N=140      PSimHit
20011008/091050  At:445       N=140      PSimHit
20011008/091050  At:585       N=952      StreamerInfo   CX =  2.66
20011008/091050  At:1537      N=64       FreeSegments
20011008/091050  At:1601      N=1        END
```

```
root [4] hit2.Dump();
```

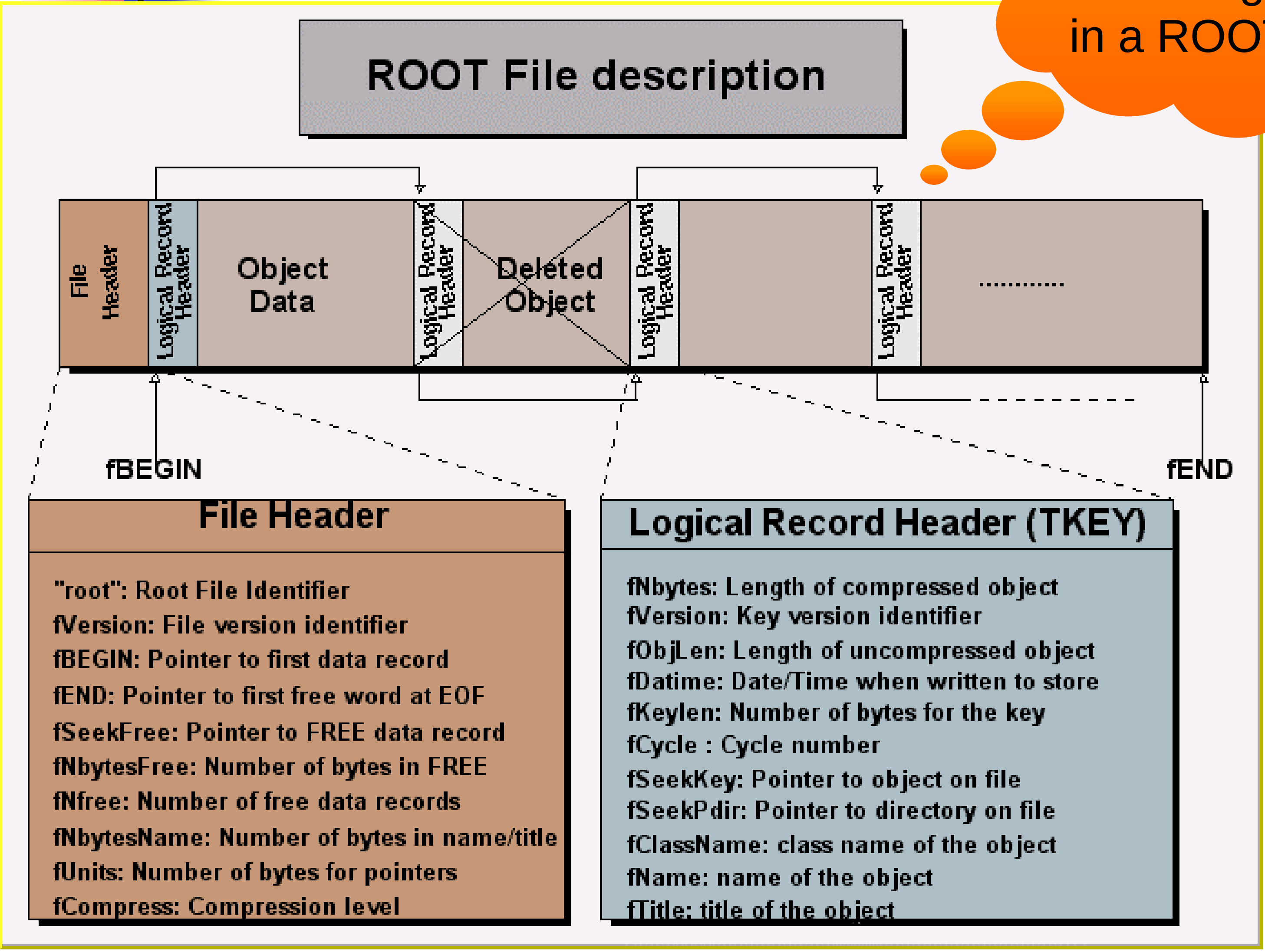
```
theEntryPoint                   ->874a01c    position
theEntryPoint.theVector         ->874a01c
theEntryPoint.theVector.theX    1
theEntryPoint.theVector.theY    2
theEntryPoint.theVector.theZ    3
theExitPoint                    ->874a030
theExitPoint.theVector          ->874a030
theExitPoint.theVector.theX     10
theExitPoint.theVector.theY     20
theExitPoint.theVector.theZ     30
thePabs                         41           momentum
theTof                          1.67e-08     Time Of Flight
theEnergyLoss                   0.00578      Energy loss
theParticleType                 12
theDetUnitId                    67
theTrackId                      1234
fUniqueID                       0            object unique identifier
fBits                           50331648     bit field status word
```

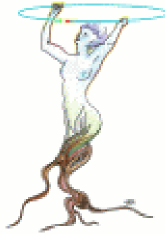
The description of
all classes
in a file
is written
in one single record
when the file is closed
StreamerInfo

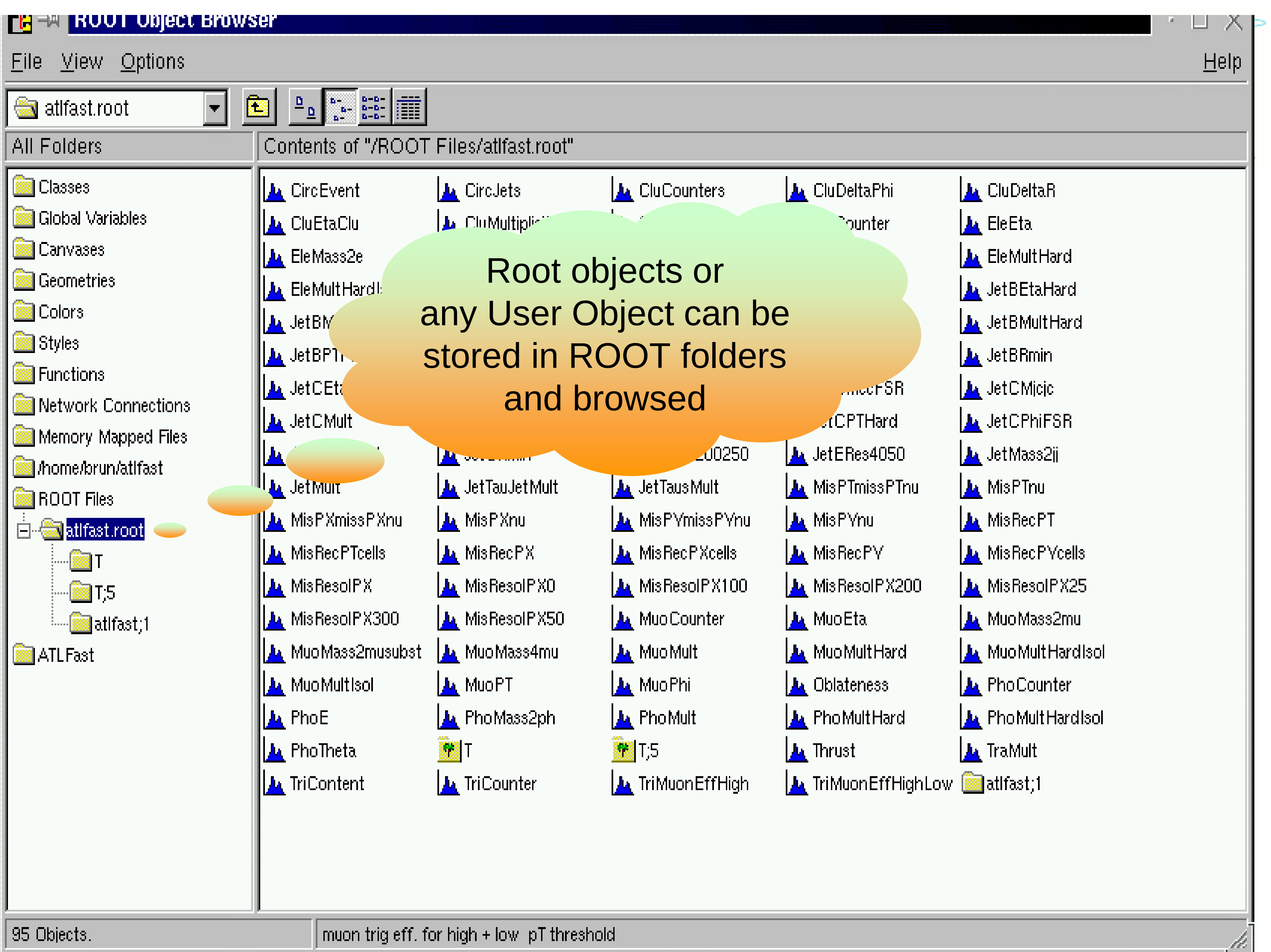
ROOT I/O -- *Sequential/Flat*



All what you need to known to navigate in a ROOT file

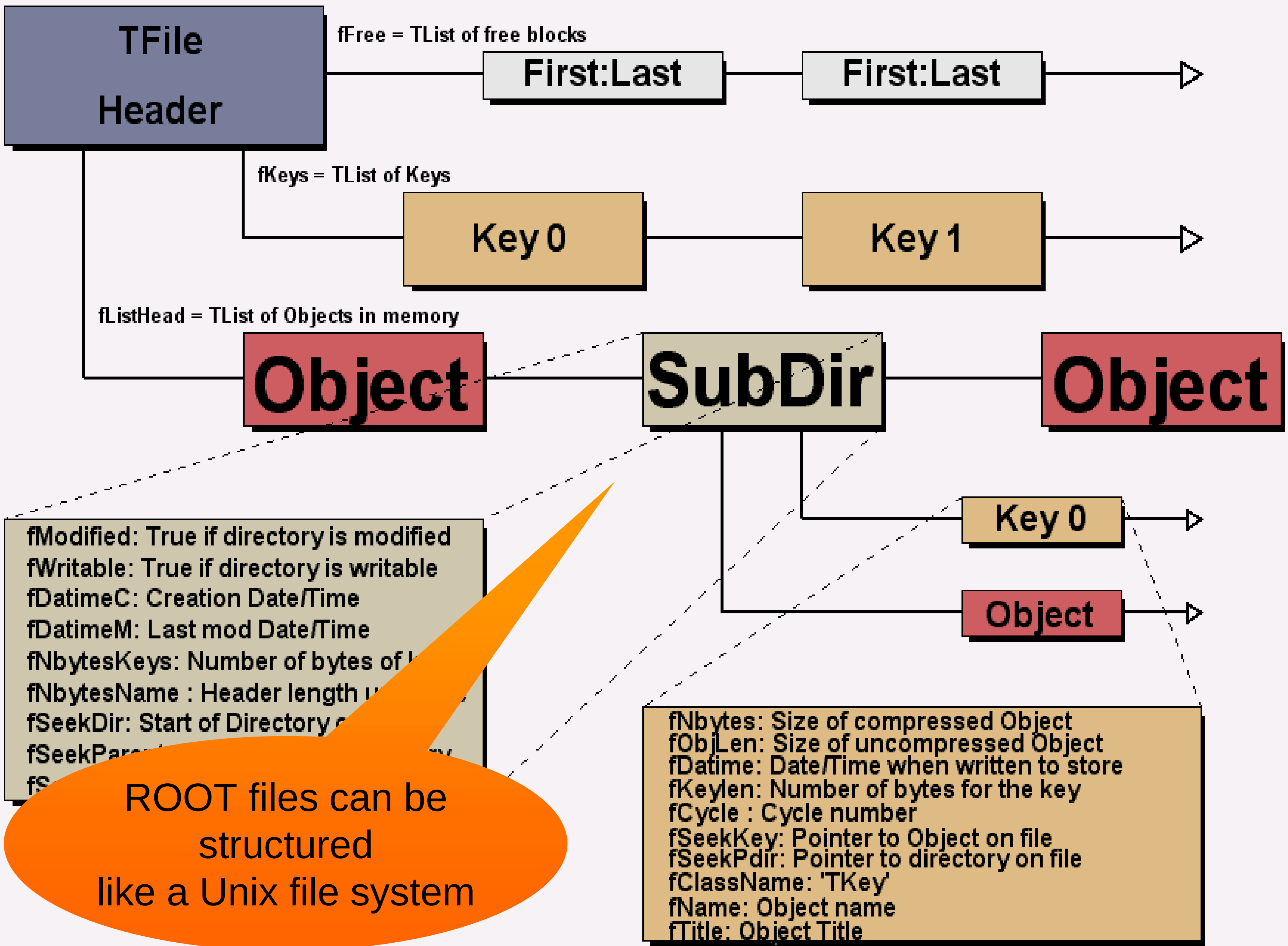


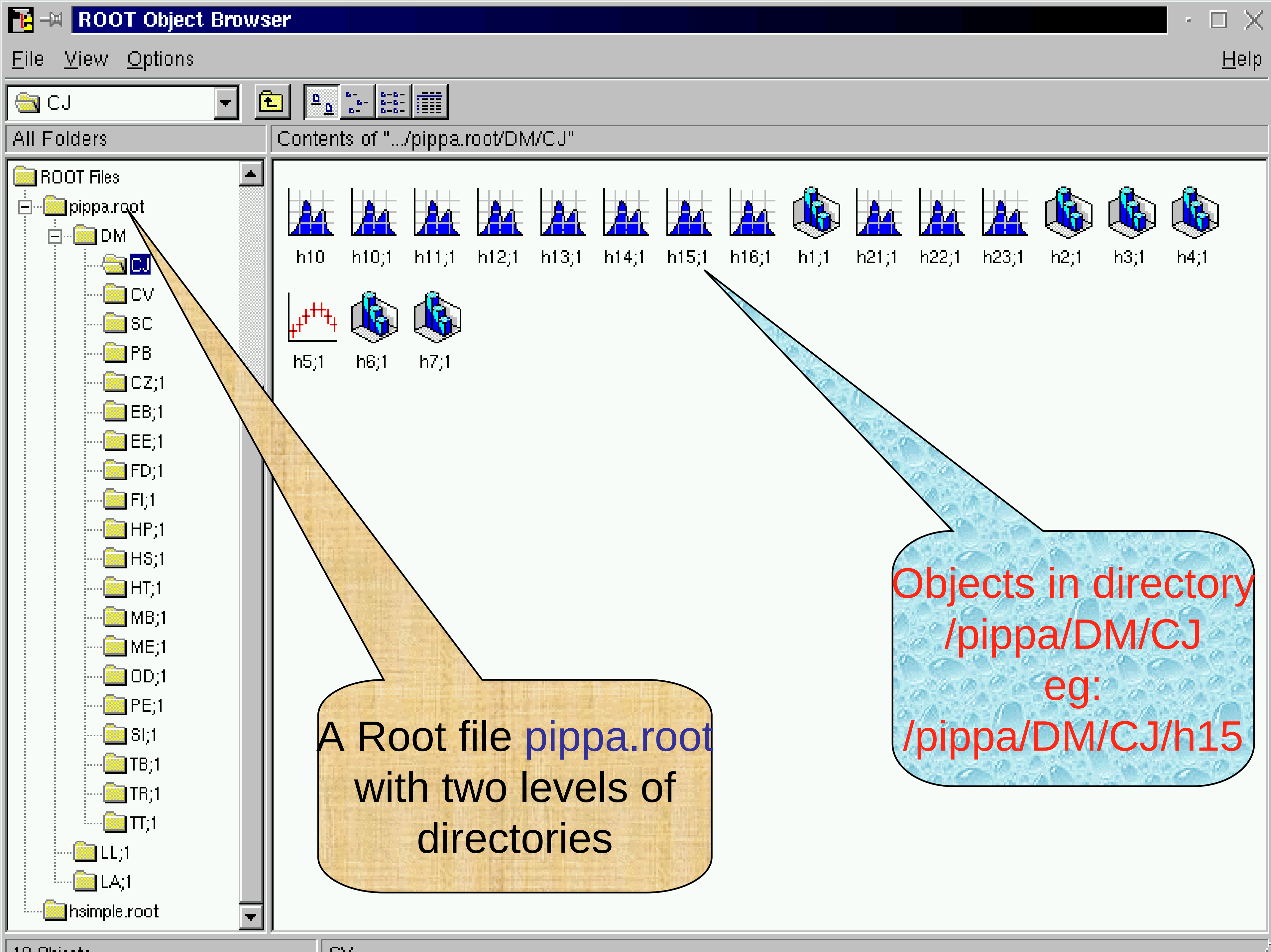




ROOT Object Browser
File View Options Help
atlfast.root
All Folders
Contents of "/ROOT Files/atlfast.root"
Classes
Global Variables
Canvases
Geometries
Colors
Styles
Functions
Network Connections
Memory Mapped Files
/home/brun/atlfast
ROOT Files
atlfast.root
T
T;5
atlfast;1
ATLFast
CircEvent
CircJets
CluCounters
CluDeltaPhi
CluDeltaR
CluEtaClu
EleEta
EleMass2e
EleMultHard
JetBEtaHard
JetBMultHard
JetBRmin
JetCMjcjc
JetCMult
JetCPhiFSR
JetERes4050
JetMass2jj
JetMult
JetTauJetMult
JetTausMult
MisPTmissPTnu
MisPTnu
MisPXmissPXnu
MisPXnu
MisPYmissPYnu
MisPYnu
MisRecPT
MisRecPTcells
MisRecPX
MisRecPXcells
MisRecPY
MisRecPYcells
MisResolPX
MisResolPX0
MisResolPX100
MisResolPX200
MisResolPX25
MisResolPX300
MisResolPX50
MuoCounter
MuoEta
MuoMass2mu
MuoMass2musubst
MuoMass4mu
MuoMult
MuoMultHard
MuoMultHardIsol
MuoMultIsol
MuoPT
MuoPhi
Oblateness
PhoCounter
PhoE
PhoMass2ph
PhoMult
PhoMultHard
PhoMultHardIsol
PhoTheta
T
T;5
Thrust
TraMult
TriContent
TriCounter
TriMuonEffHigh
TriMuonEffHighLow
atlfast;1
95 Objects.
muon trig eff. for high + low pT threshold
Root objects or
any User Object can be
stored in ROOT folders
and browsed

ROOT File/Directory/Key description




ROOT Object Browser
File View Options Help
CJ
All Folders
Contents of ".../pippa.root/DM/CJ"
ROOT Files
pippa.root
DM
CJ
CV
SC
PB
CZ;1
EB;1
EE;1
FD;1
FI;1
HP;1
HS;1
HT;1
MB;1
ME;1
OD;1
PE;1
SI;1
TB;1
TR;1
TT;1
LL;1
LA;1
hsimple.root
h10
h10;1
h11;1
h12;1
h13;1
h14;1
h15;1
h16;1
h1;1
h21;1
h22;1
h23;1
h2;1
h3;1
h4;1
h5;1
h6;1
h7;1
A Root file pippa.root with two levels of directories
Objects in directory /pippa/DM/CJ eg: /pippa/DM/CJ/h15

LAN/WAN files

See Fons talk

- Files and Directories
  - a directory holds a list of named objects
  - a file may have a hierarchy of directories (a la Unix)
  - ROOT files are machine independent
  - built-in compression
- Support for local, LAN and WAN files
  - TFile f1("myfile.root")
  - TFile f2("http://pcbrun.cern.ch/Renefile.root")
  - TFile f3("root://cdfsga.fnal.gov/bigfile.root")
  - TFile f4("rfio://alice/run678.root")

Local file

Remote file access via a Web server

Remote file access via the ROOT daemon

Access to a file on a mass store hpps, castor, via RFIO

Streaming Objects

Old Streamers in 0.90 (1996)

Evolution illustrated with the ROOT class TAxis

TAxis.h

```
class TAxis : public
TNamed,
public TAttAxis {

private:
  Int_t         fNbins;
  Float_t       fXmin;
  Float_t       fXmax;
  TArrayF       fXbins;
```

rootcint

Dict.cxx

```
void TAxis::Streamer(TBuffer &b)
{
   if (b.IsReading()) {
      Version_t v = b.ReadVersion();
      TNamed::Streamer(b);
      TAttAxis::Streamer(b);
      b >> fNbins;
      b >> fXmin;
      b >> fXmax;
      fXbins.Streamer(b);
   } else {
      b.WriteVersion(TAxis::IsA());
      TNamed::Streamer(b);
      TAttAxis::Streamer(b);
      b << fNbins;
      b << fXmin;
      b << fXmax;
      fXbins.Streamer(b);
   }
}
```

```
TBuffer b;
object.Streamer(b);
```

Old Streamers in 2.25 (1999)

```
class TAxis : public TNamed,
public TAttAxis {

private:
   Int_t        fNbins;
   Float_t      fXmin;
   Float_t      fXmax;
   TArrayF      fXbins;
   Int_t        fFirst;
   Int_t        fLast;
   TString      fTimeFormat;
   Bool_t       fTimeDisplay;
```

rootcint

```
void TAxis::Streamer(TBuffer &b) {
   UInt_t R__s, R__c;
   if (b.IsReading()) {
      Version_t v = b.ReadVersion(&R__s, &R__c);
      TNamed::Streamer(b);
      TAttAxis::Streamer(b);
      b >> fNbins;
      b >> fXmin;
      b >> fXmax;
      fXbins.Streamer(b);
      if (v > 2) {
         b >> fFirst;
         b >> fLast;
      }
      if (v > 3) {
         b >> fTimeDisplay;
         fTimeFormat.Streamer(b);
      } else {
         SetTimeFormat();
      }
      b.CheckByteCount(R__s, R__c, TAxis::IsA());
   } else {
      R__c = b.WriteVersion(TAxis::IsA(), kTRUE);
      TNamed::Streamer(b);
      TAttAxis::Streamer(b);
      b << fNbins;
      b << fXmin;
      b << fXmax;
      fXbins.Streamer(b);
      b << fFirst;
      b << fLast;
      b << fTimeDisplay;
      fTimeFormat.Streamer(b);
      b.SetByteCount(R__c, kTRUE);
   }
}
```

Problems with Old Streamers

- Experience in several large experiments has shown that a system based only on automatic code generation with no support for schema evolution is not a long term solution. A huge maintenance problem.
- In a system with several hundred (thousand) classes and as many users, it is difficult to maintain coherent shared libs to support all possible combinations when accessing collections of old data sets.
- A few attempts (eg in STAR) to support automatic schema evolution seen as a progress, but not sufficient.
- We have seen a rapidly growing request for reading data sets without having the original classes.
- Backward compatibility (reading an old data set with new classes) is a must. Forward compatibility (reading a new data set with old classes) also a must.

The ROOT solution

- Minimize reliance on generated code.
- Exploit the powerful CINT Object Dictionary
- Make the process as automatic as possible and as simple as possible.
- Be as efficient as with the generated code.
- Self-describing data sets.
- Come with a solution that does not prevent the move to another language in the future.
- Back compatibility with the original system.
  - Like upgrading the engine in a running car

Implementing all these features was a non trivial exercise and a lot of work

Thanks to our huge users base for providing many use cases and testing

New Streamers in 3.00

```
class TAxis : public TNamed,
public TAttAxis {

private:
   Int_t        fNbins;
   Double_t     fXmin;
   Double_t     fXmax;
   TArrayD      fXbins;
   Char_t      *fXlabels;     //!
   Int_t        fFirst;
   Int_t        fLast;
   TString      fTimeFormat;
   Bool_t       fTimeDisplay;
   TObject     *fParent;      //!
```

rootcint

```
void TAxis::Streamer(TBuffer &b)
{
   // Stream an object of class TAxis.

   if (b.IsReading())
      TAxis::Class()->ReadBuffer(b, this);
   else
      TAxis::Class()->WriteBuffer(b,this);
}
```

Support for more complex C++

```
enum {kSize=10};
   char             fType[20];      //array of 20 chars
   Int_t            fNtrack;        //number of tracks
   Int_t            fNvertex;       //number of vertices
   Int_t            fX[kSize];      //an array where dimension is an enum
   UInt_t           fFlag;          //bit pattern event flag
   Float_t          fMatrix[4][4]; //a two-dim array
   Float_t         *fDistance;      //[fNvertex] array of floats of length fNvertex
   Double_t         fTemperature;  //event temperature
   TString         *fTstringp;      //[fNvertex] array of TString
   TString          fNames[12];     //array of TString
   TAxis            fXaxis;         //example of class derived from TObject
   TAxis            fYaxis[3];      //array of objects
   TAxis           *fVaxis[3];      //pointer to an array of TAxis
   TAxis           *fPaxis;         //[fNvertex] array of TAxis of length fNvertex
   TAxis          **fQaxis;         //[fNvertex] array of pointers to TAxis objects
   TDatime          fDatime;        //date and time
   EventHeader      fEvtHdr;        //example of class not derived from TObject
   TObjArray        fObjArray;      //An object array of TObject*
   TClonesArray    *fTracks;        //-> array of tracks
   TH1F            *fH;             //-> pointer to an histogram
   TArrayF          fArrayF;        //an array of floats
   TArrayI         *fArrayI;        //a pointer to an array of integers
………………..(see next)
```

Support for STL

```
vector<int>            fVectorint;       //STL vector on ints
vector<short>          fVectorshort;     //STL vector of shorts
vector<double>         fVectorD[4];      //array of STL vectors of doubles
vector<TLine>          fVectorTLine;     //|| STL vector of TLine objects
vector<TObject>       *fVectorTobject;   //|| pointer to an STL vector
vector<TNamed>        *fVectorTnamed[6]; //|| array of pointers to STL vectors
deque<TAttLine>        fDeque;           //STL deque
list<const TObject*>   fVectorTobjectp;  //STL list of pointers to objects
list<string>          *fListString;      //STL list of strings
list<string *>         fListStringp;     //STL list of pointers to strings
map<TNamed*,int>       fMapTNamedp;      //STL map

map<TString,TList*>    fMapList;         //STL map
map<TAxis*,int>       *fMapTAxisp;       //pointer to STL map
set<TAxis*>            fSetTAxis;        //STL set
set<TAxis*>           *fSetTAxisp;       //pointer to STL set
multimap<TNamed*,int>  fMultiMapTNamedp; //STL multimap
multiset<TAxis*>      *fMultiSetTAxisp;  //pointer to STL multiset
string                 fString;          //C++ standard string
string                *fStringp;         //pointer to standard C++ string
UShortVector           fUshort;          //class with an STL vector as base class
```

Complex STL use not supported

```
vector<vector<TAxis *> >  fVectAxis;    //!STL vector of vectors of TAxis*
map<string,vector<int> >  fMapString;   //!STL map of string/vector
deque<pair<float,float> > fDequePair;   //!STL deque of pair
```

Use a custom Streamer
for these complex cases

The ROOT Collection Classes

All ROOT collections support Polymorphism

- -TCollection (abstract base class)
- -TSeqCollection, TList, THashList
- -TMap, TExMap
- -TObjArray

```
Tbuffer b;
collection.Streamer(b);
```

TClonesArray is a specialized collection for arrays of objects of the same class.

It minimizes the overhead due to new/delete. Much more efficient than STL for I/O (see next slides)

TRefArray is an optimized collection for persistent reference pointers.

See example Event

New in 3.02

The Test suite "bench"

(example on fcdfsgi2 with KAI compiler)

- Test performance of STL vector of objects, vectors of pointers and same with a TClonesArray of TObjHit deriving from THit

Time to fill the structures (seconds)	Reference	cx	Reference
vector<THit>	2.16	1.91	4.57	4.57
vector<THit*>	2.36	1.86	4.56	4.57
TClonesArray(TObjHit)	1.98	1.62	6.77	6.76
TClonesArray(TObjHit) split	1.98	1.62	6.76	6.75

Size of file in bytes	comp 0	Reference	comp 1	Reference
vector<THit>	42053031	42053031	9213642	9213459
vector<THit*>	42079941	42079941	9220556	9215935
TClonesArray(TObjHit)	39807325	39807325	5878130	5892837
TClonesArray(TObjHit) split	39807325	39807325	5890726	5901163

Time to write in seconds	comp 0	Reference	comp 1	Reference
vector<THit>	2.63	1.74	9.34	9.58
vector<THit*>	2.47	1.80	9.44	9.62
TClonesArray(TObjHit)	1.33	1.60	5.45	7.32
TClonesArray(TObjHit) split	1.23	1.51	5.46	6.18

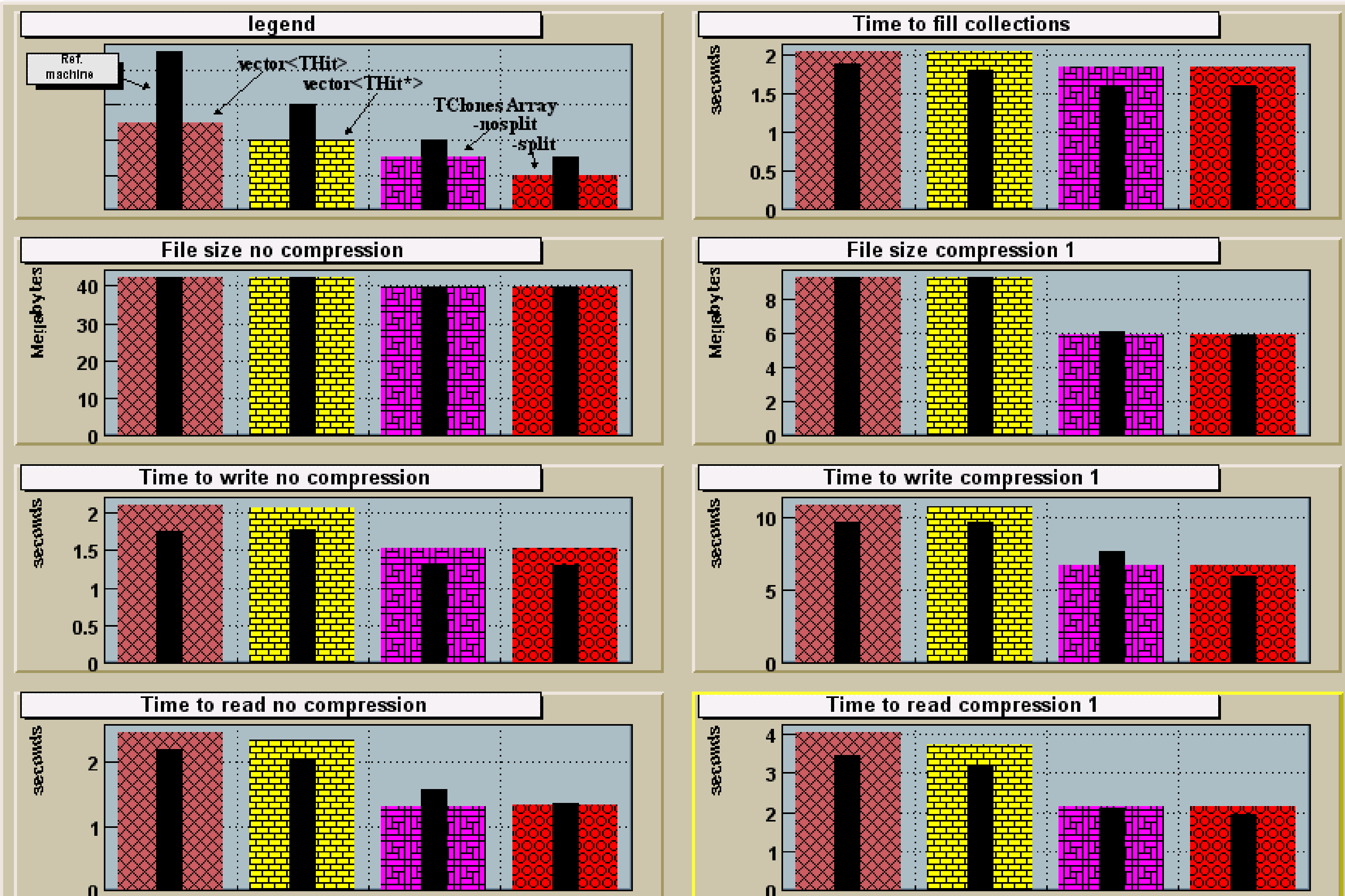
Time to read in seconds	comp 0	Reference	comp 1	Reference
vector<THit>	3.03	2.29	4.24	3.67
vector<THit*>	3.01	2.10	4.28	3.27
TClonesArray(TObjHit)	1.34	1.53	1.88	2.14
TClonesArray(TObjHit) split	1.35	1.35	1.88	1.94
Total CPU time	82.14	76.33		
Estimated ROOTMARKS	185.85	200.00		

Better compression with TClonesArray

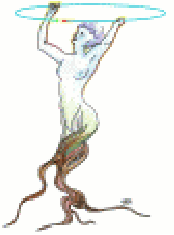
Better write with TClonesArray

Much better read with TClonesArray

Comparing STL vector with TClonesArray: Root 3.01/03
Linux pcnotebrun 2.2.19 #1 Tue May 15 12:33:56 MEST 2001 i6
Reference machine pcnotebrun.cern.ch RedHat Linux 6.1
(Pentium III 650 Mhz 256 Mbytes RAM, IDE disk)
(send your results to rootdev@root.cern.ch)



Self-Describing file



```
Root > TFile f("demo1.root");
Root > f.ShowStreamerInfo();

StreamerInfo for class: PSimHit, version=1
  BASE          TObject          offset=  0 type=66 Basic ROOT object
  Local3DPoint  theEntryPoint    offset=  0 type=62 position
  Local3DPoint  theExitPoint     offset=  0 type=62
  float         thePabs          offset=  0 type= 5 momentum
  float         theTof           offset=  0 type= 5 Time Of Flight
  float         theEnergyLoss    offset=  0 type= 5 Energy loss
  int           theParticleType  offset=  0 type= 3
  int           theDetUnitId     offset=  0 type= 3
  unsigned int  theTrackId       offset=  0 type=13

StreamerInfo for class: Point3DBase<float,LocalTag>, version=1
  BASE          PV3DBase<float,PointTag,LocalTag> offset=  0 type= 0

StreamerInfo for class: PV3DBase<float,PointTag,LocalTag>, version=1
  Basic3DVector<float>theVector        offset=  0 type=62

StreamerInfo for class: Basic3DVector<float>, version=1
  float         theX             offset=  0 type= 5
  float         theY             offset=  0 type= 5
  float         theZ             offset=  0 type= 5
```

Self-describing files

- Dictionary for persistent classes written to the file when closing the file.
- ROOT files can be read by foreign readers (eg JavaRoot (Tony Johnson)
- Support for Backward and Forward compatibility
- Files created in 2003 must be readable in 2015
- Classes (data objects) for all objects in a file can be regenerated via TFile::MakeProject

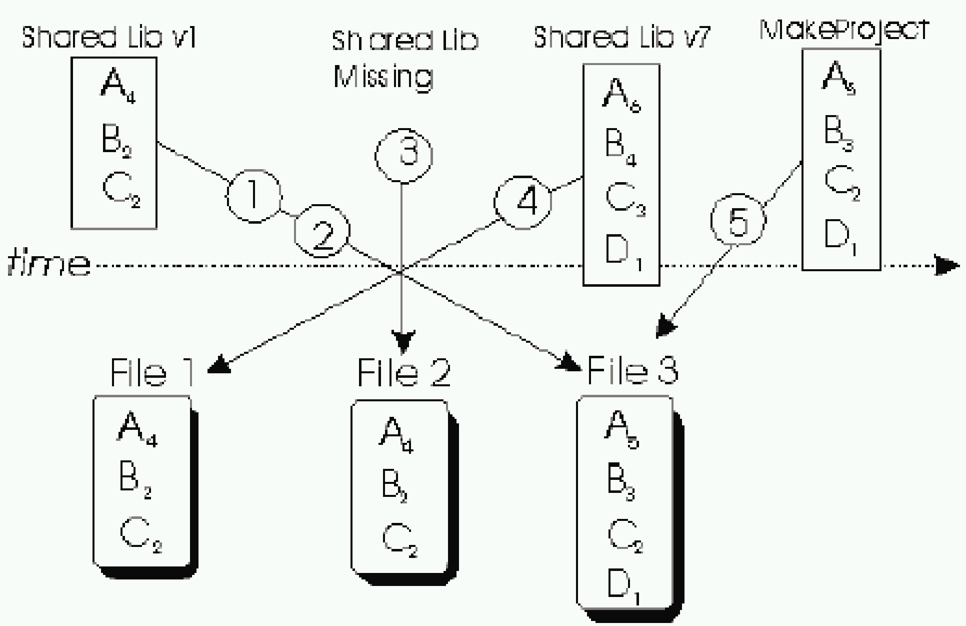
```
Root >TFile f(“demo.root”);
Root > f.MakeProject(“dir”,”*”,”new++”);
```

Showing classes in a file TFile::ShowStreamerInfo

```
Root > f.ShowStreamerInfo()
```

```
StreamerInfo for class: ATLFMuon, version=1
  BASE          TObject        offset=  0 type=66 Basic ROOT object
  BASE          TAtt3D         offset=  0 type= 0 3D attributes
  Int_t         m_KFcode       offset=  0 type= 3 Muon KF-code
  Int_t         m_MCParticle   offset=  0 type= 3 Muon position in MCParticles list
  Int_t         m_KFmother     offset=  0 type= 3 Muon mother KF-code
  Int_t         m_UseFlag      offset=  0 type= 3 Muon energy usage flag (0 for used in clusters)
  Int_t         m_Isolated     offset=  0 type= 3 Muon isolation (1 for isolated)
  Float_t       m_Eta          offset=  0 type= 5 Eta coordinate
  Float_t       m_Phi          offset=  0 type= 5 Phi coordinate
  Float_t       m_PT           offset=  0 type= 5 Transverse energy
  Int_t         m_Trigger      offset=  0 type= 3 Result of trigger

StreamerInfo for class: ATLFElectron, version=1
  BASE          TObject        offset=  0 type=66 Basic ROOT object
  BASE          TAtt3D         offset=  0 type= 0 3D attributes
  Int_t         m_KFcode       offset=  0 type= 3 Electron KF-code
  Int_t         m_MCParticle   offset=  0 type= 3 Electron position in MCParticles list
  Int_t         m_KFmother     offset=  0 type= 3 Electron mother KF-code
  Float_t       m_Eta          offset=  0 type= 5 Eta coordinate
  Float_t       m_Phi          offset=  0 type= 5 Phi coordinate
  Float_t       m_PT           offset=  0 type= 5 Transverse energy

StreamerInfo for class: ATLFPhoton, version=1
  BASE          TObject        offset=  0 type=66 Basic ROOT object
  BASE          TAtt3D         offset=  0 type= 0 3D attributes
  Int_t         m_KFcode       offset=  0 type= 3 Photon KF-code
  Int_t         m_MCParticle   offset=  0 type= 3 Photon position in MCParticles list
  Int_t         m_KFmother     offset=  0 type= 3 Photon mother KF-code
  Float_t       m_Eta          offset=  0 type= 5 Eta coordinate
  Float_t       m_Phi          offset=  0 type= 5 Phi coordinate
  Float_t       m_PT           offset=  0 type= 5 Transverse energy

StreamerInfo for class: ATLFJet, version=1
  BASE          TObject        offset=  0 type=66 Basic ROOT object
  BASE          TAtt3D         offset=  0 type= 0 3D attributes
  Int_t         m_KFcode       offset=  0 type= 3 Jet KF-code
  Int_t         m_Ncells       offset=  0 type= 3 Number of cells used for reconstruction
  Int_t         m_Nparticles   offset=  0 type= 3 Number of particles assigned to jet
  Int_t         m_Part         offset=  0 type= 3 Position in MCParticle list of matching b-quark/c-qu
  Float_t       m_Eta0         offset=  0 type= 5 Eta position of initiator cell
  Float_t       m_Phi0         offset=  0 type= 5 Phi position of initiator cell
  Float_t       m_Eta          offset=  0 type= 5 Eta  of jet bary-center
  Float_t       m_Phi          offset=  0 type= 5 Phi  of jet bary-center
  Float_t       m_PT           offset=  0 type= 5 Transverse momentum of jet

....
```

Automatic Schema Evolution



1) An old version of a shared library and a file with new class definitions. This can be the case when someone has not updated the library and is reading a new file.

2) Reading a file with a shared library that is missing a class definition (i.e. missing class D).

3) Reading a file without any class definitions. This can be the case where the class definition is lost, or unavailable.

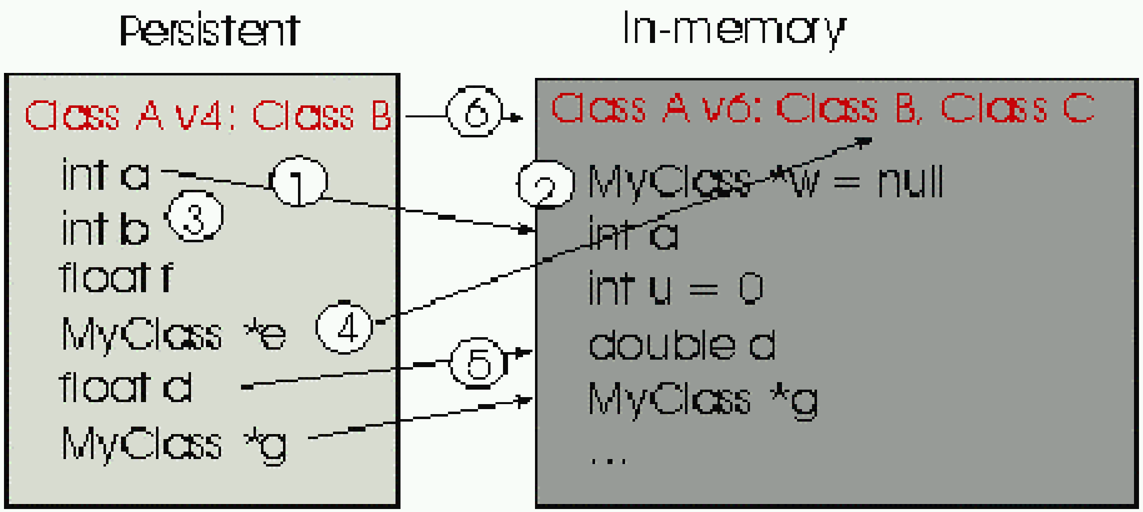
4) The current version of a shared library and an old file with old class versions (backward compatibility). This is often the case when reading old data.

5) Reading a file with a shared library built with `MakeProject`. This is the case when someone has already read the data without a shared library and has used ROOT's `MakeProject` feature to reconstruct the class definitions and shared library (`MakeProject` is explained in detail later on).

Auto Schema Evolution (2)

In case of a mismatch between the in-memory version and the persistent version of a class, ROOT maps the persistent one to the one in memory. This allows you to change the class definition at will, for example:

1) Change the order of data members in the class.
2) Add new data members. By default the value of the missing member will be 0 or in case of an object it will be set to null.
3) Remove data members.
4) Move a data member to a base class or vice –versa.
5) Change the type of a member if it is a simple type or a pointer to a simple type. If a loss of precision occurs, a warning is given.
6) Add or remove a base class



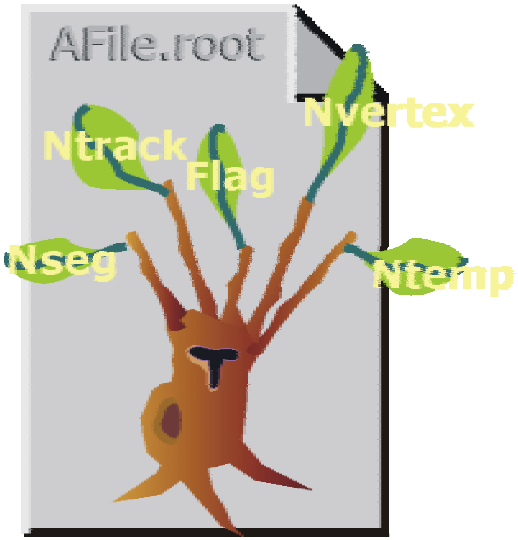
ROOT Trees

Ntuples and Trees

- Ntuples
  - support PAW-like ntuples and functions
  - PAW ntuples/histograms can be imported
- Trees
  - Extension of Ntuples for Objects
  - Collection of branches (branch has its own buffer)
  - Can input partial Event
  - Can have several Trees in parallel
- Chains = collections of Trees

Why Trees ?

- Any object deriving from TObject can be written to a file with an associated key with object.Write()
- However each key has an overhead in the directory structure in memory (about 60 bytes). Object.Write is very convenient for objects like histograms, detector objects, calibrations, but not for event objects.



Why Trees ?

- Trees have been designed to support very large collections of objects. The overhead in memory is in general less than 4 bytes per entry.
- Trees allow direct and random access to any entry (sequential access is the best)
- Trees have branches and leaves. One can read a subset of all branches. This can speed-up considerably the data analysis processes.

Why Trees ?

- PAW ntuples are a special case of Trees.
- Trees are designed to work with complex event objects.
- High level functions like TTree::Draw loop on all entries with selection expressions.
- Trees can be browsed via TBrowser
- Trees can be analized via TTreeViewer

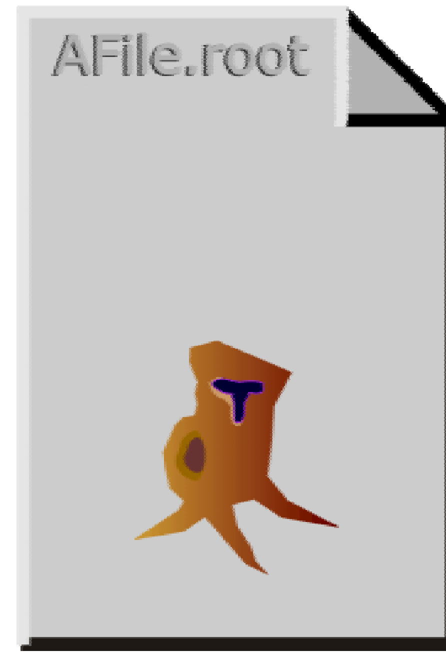
The PROOF system is designed to process chains of Trees in parallel in a GRID environment

Create a TTree Object

A tree is a list of branches.

The TTree Constructor:

- Tree Name (e.g. "myTree")
- Tree Title

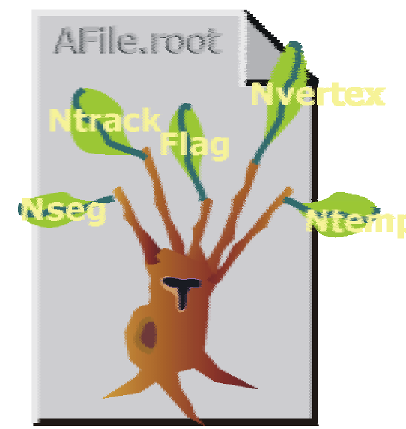



```
TTree *tree = new TTree("T","A ROOT tree");
```

Adding a Branch

Many Branch constructors
Only a few shown here

- Branch name
- Class name
- Address of the pointer to the Object (descendant of TObject)
- Buffer size (default = 32,000)
- Split level (default = 1)

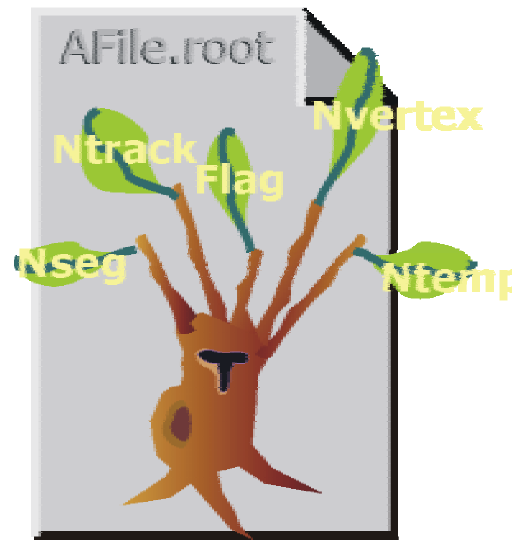



```
Event *event = new  Event();
myTree->Branch(”eBranch","Event",&event,64000,1);
```

Splitting a Branch

Setting the split level (default = 1)

Split level = 0



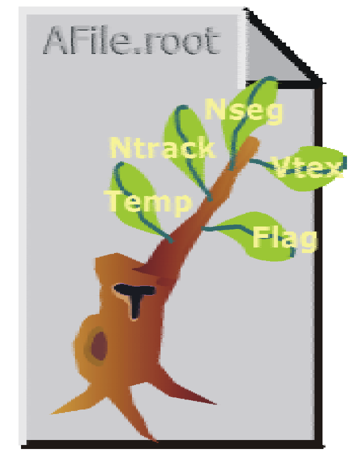
Split level = 1

Example:

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

Adding 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 (default = 1)

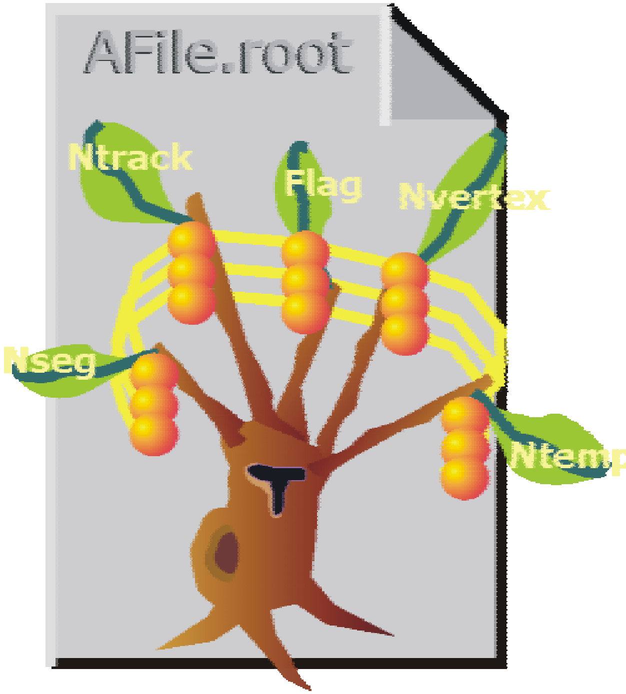
Example:

```
tree->Branch( "Track_B",&Track,64000,1);
```

Filling the Tree

- Create a for loop
- Create Event objects.
- Call the Fill method for the tree.

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

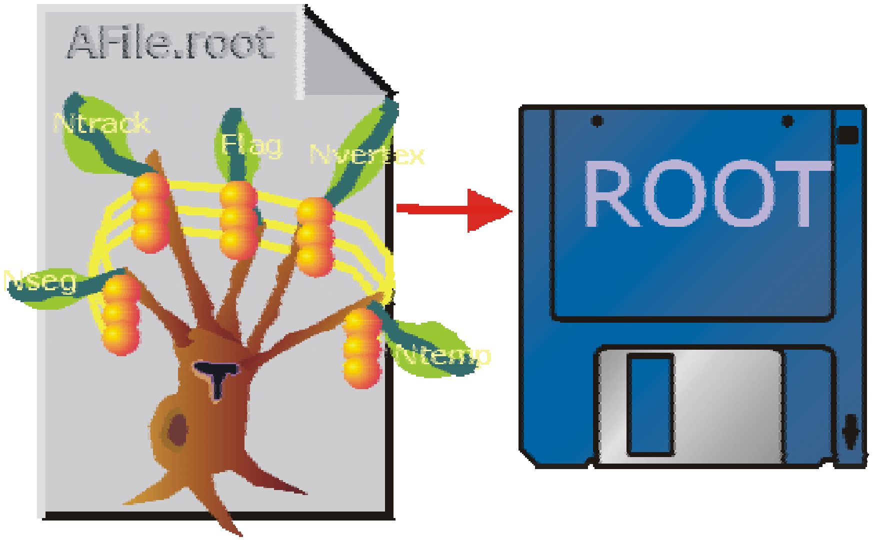


Write the Tree header

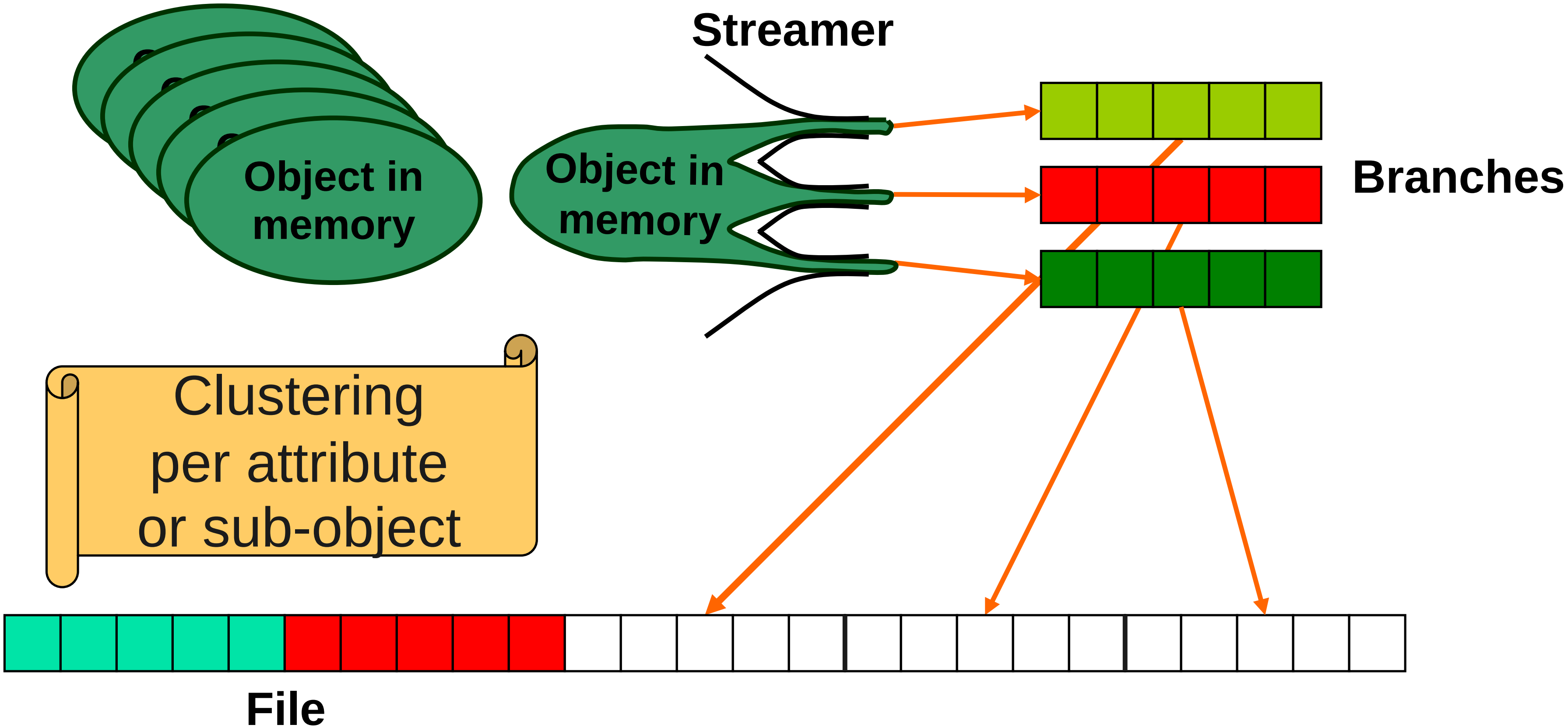
The Tree header contains a description of the Tree

- It owns the collection of branches
- Each branch has a buffer (TBasket) partially filled
- TTree::Write writes one single record on the file

```
tree->Write();
```

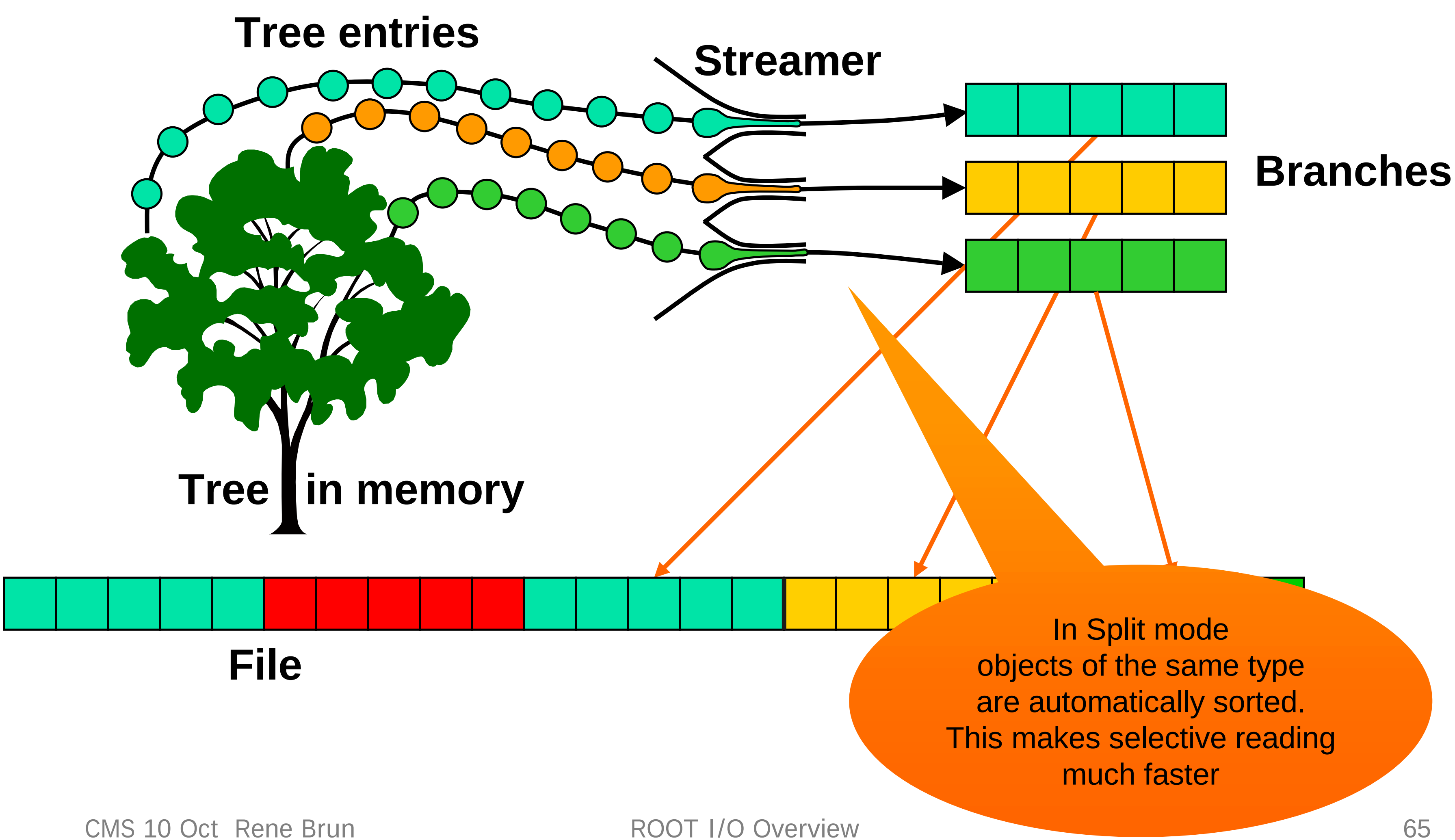


ROOT I/O -- *Split/Cluster*

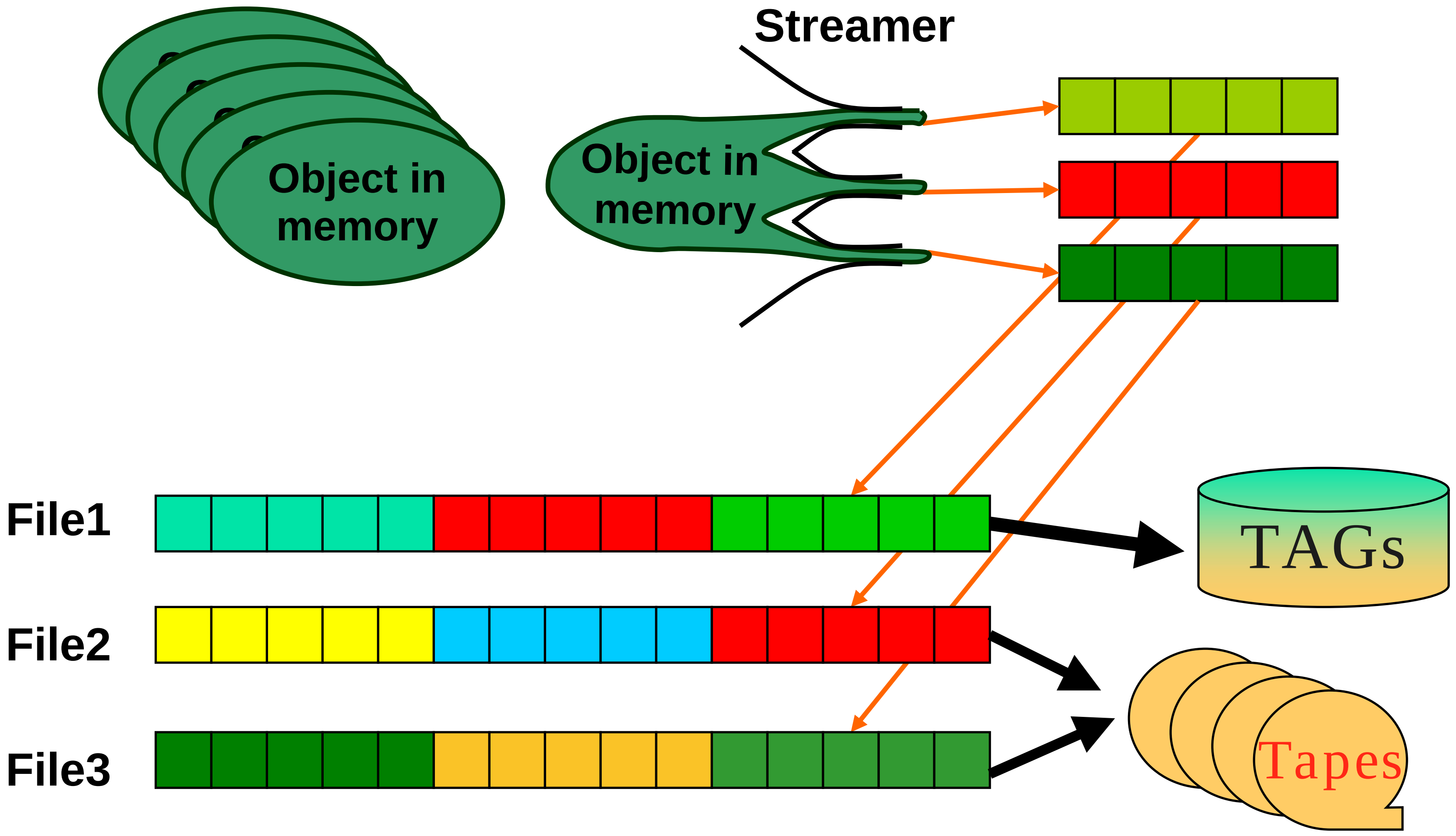


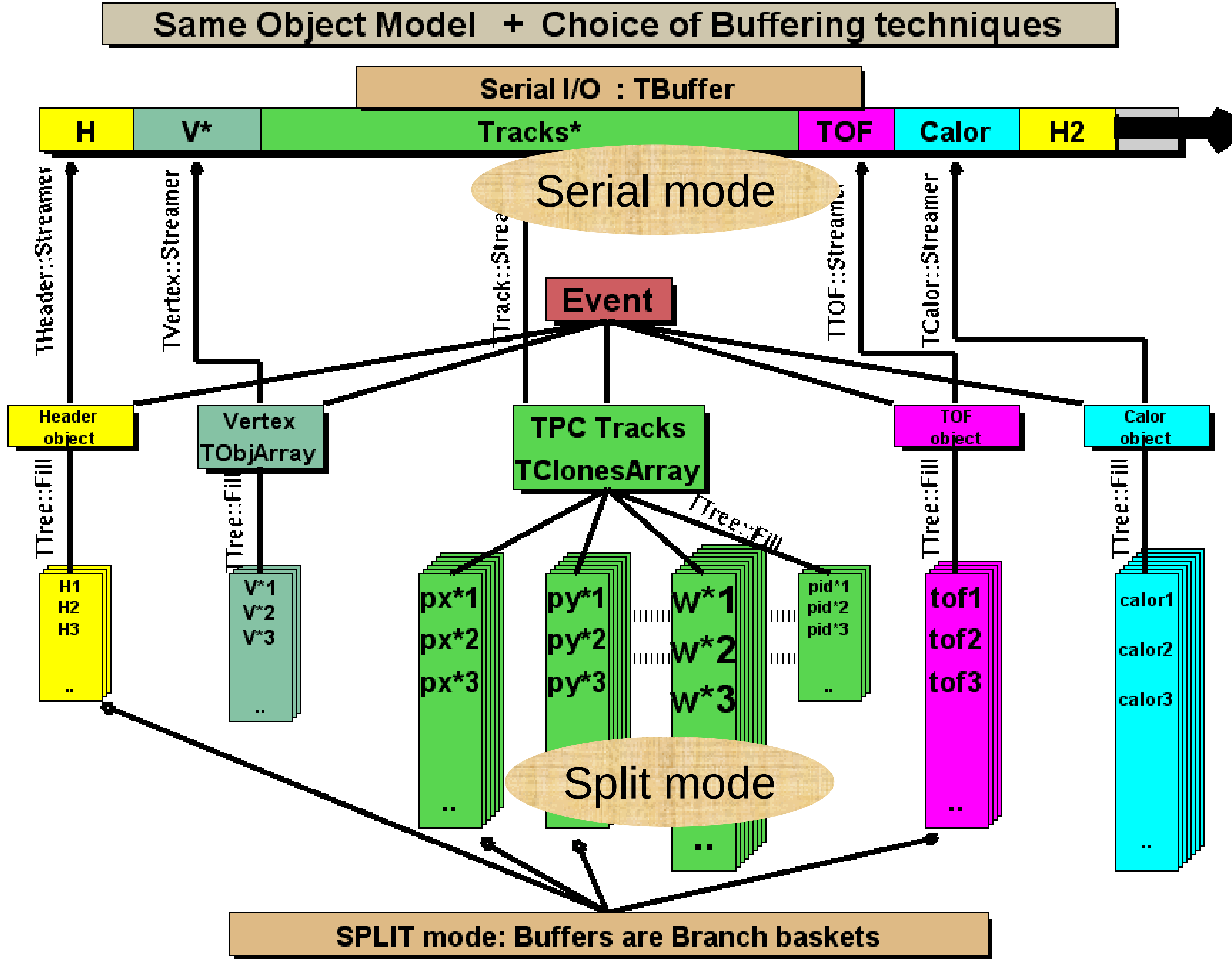
ROOT I/O -- *Split/Cluster*

Tree version

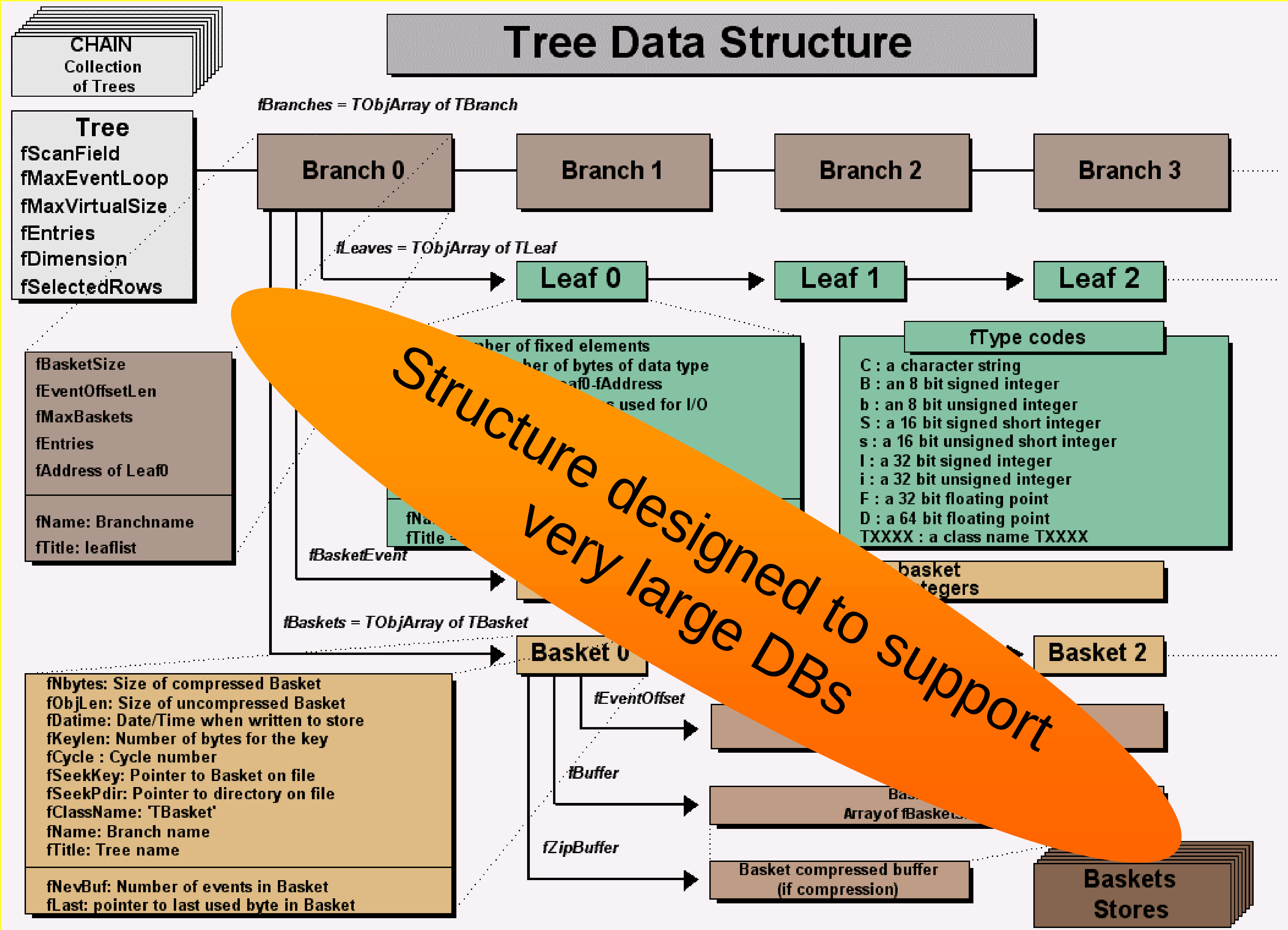


ROOT I/O - *Split - multifile*





/user/brun/root/slides/chep97/iomode.C


Tree Data Structure
CHAIN
Collection
of Trees
Tree
fScanField
fMaxEventLoop
fMaxVirtualSize
fEntries
fDimension
fSelectedRows
fBranches = TObjArray of TBranch
Branch 0
Branch 1
Branch 2
Branch 3
fLeaves = TObjArray of TLeaf
Leaf 0
Leaf 1
Leaf 2
fBasketSize
fEventOffsetLen
fMaxBaskets
fEntries
fAddress of Leaf0
fName: Branchname
fTitle: leaflist
fType codes
C : a character string
B : an 8 bit signed integer
b : an 8 bit unsigned integer
S : a 16 bit signed short integer
s : a 16 bit unsigned short integer
I : a 32 bit signed integer
i : a 32 bit unsigned integer
F : a 32 bit floating point
D : a 64 bit floating point
TXXXX : a class name TXXXX
fBasketEvent
fBaskets = TObjArray of TBasket
Basket 2
fNbytes: Size of compressed Basket
fObjLen: Size of uncompressed Basket
fDatime: Date/Time when written to store
fKeylen: Number of bytes for the key
fCycle : Cycle number
fSeekKey: Pointer to Basket on file
fSeekPdir: Pointer to directory on file
fClassName: 'TBasket'
fName: Branch name
fTitle: Tree name
fNevBuf: Number of events in Basket
fLast: pointer to last used byte in Basket
fEventOffset
fBuffer
fZipBuffer
Basket compressed buffer
(if compression)
Baskets
Stores
Structure designed to support
very large DBs

The Event class

```
class Event : public TObject {

private:
   char            fType[20];          //event type
   Int_t           fNtrack;            //Number of tracks
   Int_t           fNseg;              //Number of track segments
   Int_t           fNvertex;
   UInt_t          fFlag;
   Float_t         fTemperature;
   Int_t           fMeasures[10];
   Float_t         fMatrix[4][4];
   Float_t        *fClosestDistance;   //[fNvertex]
   EventHeader     fEvtHdr;
   TClonesArray   *fTracks;            //->array with all tracks
   TRefArray      *fHighPt;            //array of High Pt tracks only
   TRefArray      *fMuons;             //array of Muon tracks only
   TRef            fLastTrack;         //reference pointer to last track
   TH1F           *fH;                 //->
```

```
class EventHeader {

private:
   Int_t   fEvtNum;
   Int_t   fRun;
   Int_t   fDate;
```

See $ROOTSYS/test/Event.h

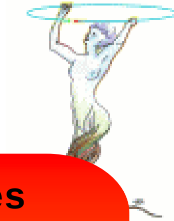
The Track class

```
class Track : public TObject {

private:
   Float_t      fPx;           //X component of the momentum
   Float_t      fPy;           //Y component of the momentum
   Float_t      fPz;           //Z component of the momentum
   Float_t      fRandom;       //A random track quantity
   Float_t      fMass2;        //The mass square of this particle
   Float_t      fBx;           //X intercept at the vertex
   Float_t      fBy;           //Y intercept at the vertex
   Float_t      fMeanCharge;   //Mean charge deposition of all hits
   Float_t      fXfirst;       //X coordinate of the first point
   Float_t      fXlast;        //X coordinate of the last point
   Float_t      fYfirst;       //Y coordinate of the first point
   Float_t      fYlast;        //Y coordinate of the last point
   Float_t      fZfirst;       //Z coordinate of the first point
   Float_t      fZlast;        //Z coordinate of the last point
   Float_t      fCharge;       //Charge of this track
   Float_t      fVertex[3];    //Track vertex position
   Int_t        fNpoint;       //Number of points for this track
   Short_t      fValid;        //Validity criterion
```

Event Builder

```
void Event::Build(Int_t ev, Int_ntrack, Float_t ptmin) {

  Clear();
  ………..
  for (Int_t t = 0; t < ntrack; t++) AddTrack(random,ptmin);
}

Track *Event::AddTrack(Float_t random, Float_t ptmin)
{
    // Add a new track to the list of tracks for this event.
   // To avoid calling the very time consuming operator new for each track,
   // the standard but not well know C++ operator "new with placement"
   // is called. If tracks[i] is 0, a new Track object will be created
   // otherwise the previous Track[i] will be overwritten.

    TClonesArray &tracks = *fTracks;
    Track *track = new(tracks[fNtrack++]) Track(random);
    //Save reference to last Track in the collection of Tracks
    fLastTrack = track;
    //Save reference in fHighPt if track is a high Pt track
    if (track->GetPt() > ptmin)   fHighPt->Add(track);
    //Save reference in fMuons if track is a muon candidate
    if (track->GetMass2() < 0.11) fMuons->Add(track);
    return track;
}
```

Tree example Event (write)



```
void demoe(int nevents) {
   //load shared lib with the Event class
   gSystem->Load("$ROOTSYS/test/libEvent");

   //create a new ROOT file
   TFile f("demoe.root",”new");

   //Create a ROOT Tree with one single top level branch
   int split = 99;  //try also split=1 and split=0
   int bufsize = 16000;
   Event *event = new Event;
   TTree T("T","Event demo tree");
   T.Branch("event","Event",&event,bufsize,split);

   //Build Event in a loop and fill the Tree
   for (int i=0;i<nevents;i++) {
      event->Build(i);
      T.Fill();
   }

   T.Print();  //Print Tree statistics
   T.Write();  //Write Tree header to the file
}
```

All the examples
can be executed
with CINT
or the compiler

```
root > .x demoe.C
root > .x demoe.C++
```

Tree example Event (read 1)

```
void demoer() {
   //load shared lib with the Event class
   gSystem->Load("$ROOTSYS/test/libEvent");

   //connect ROOT file
   TFile *f = new TFile("demoe.root");

   //Read Tree header and set top branch address
   Event *event = 0;
   TTree *T = (TTree*)f->Get("T");
   T->SetBranchAddress("event",&event);

   //Loop on events and fill an histogram
   TH1F *h = new TH1F("hntrack","Number of tracks",100,580,620);
   int nevents = (int)T->GetEntries();
   for (int i=0;i<nevents;i++) {
      T->GetEntry(i);
      h->Fill(event->GetNtrack());
   }

   h->Draw();
}
```

Rebuild the full event in memory

Tree example Event (read 2)

```
void demoer2() {
   //load shared lib with the Event class
   gSystem->Load("$ROOTSYS/test/libEvent");

   //connect ROOT file
   TFile *f = new TFile("demoe.root");

   //Read Tree header and set top branch address
   Event *event = 0;
   TTree *T = (TTree*)f->Get("T");
   T->SetBranchAddress("event",&event);
   Tbranch *bntrack = T->GetBranch(“fNtrack”);

   //Loop on events and fill an histogram
   TH1F *h = new TH1F("hntrack","Number of tracks",100,580,620);
   int nevents = (int)T->GetEntries();
   for (int i=0;i<nevents;i++) {
      bntrack->GetEntry(i);
      h->Fill(event->GetNtrack());
   }

   h->Draw();
}
```

Read only
one branch
Much faster !

Tree example Event (read 3)

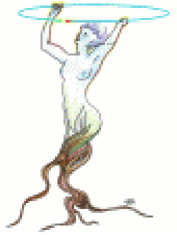
```
void demoer3() {
   //load shared lib with the Event class
   gSystem->Load("$ROOTSYS/test/libEvent");

   //connect ROOT file
   TFile *f = new TFile("demoe.root");

   //Read Tree header
   TTree *T = (TTree*)f->Get("T");

   //Histogram number of tracks via the TreePlayer
   T->Draw("event->GetNtrack()");
}
```

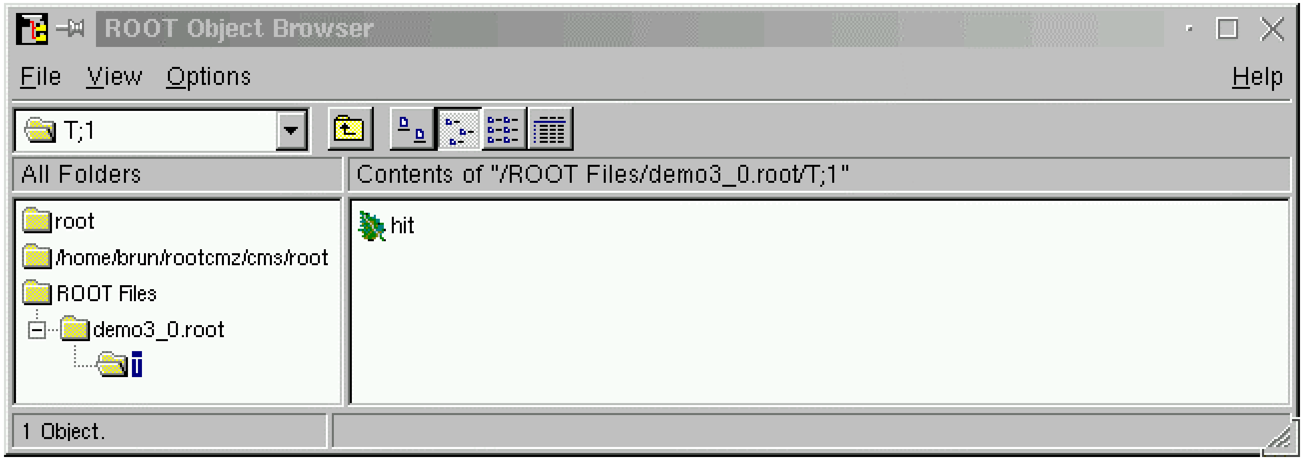
Writing CMS PSimHit in a Tree

```
void demo3() {
   //create a new ROOT file
   TFile f("demo3.root","recreate");

   //Create a ROOT Tree with one single top level branch
   int split = 99;  //you can try split=1 and split=0
   int bufsize = 16000;
   PSimHit *hit = 0;
   TTree T("T","CMS demo tree");
   T.Branch("hit","PSimHit",&hit,bufsize,split);

   //Create hits in a loop and fill the Tree
   TRandom r;
   for (int i=0;i<50000;i++) {
      delete hit;
      Local3DPoint pentry(r.Gaus(0,1), r.Gaus(0,1), r.Gaus(0,10));
      Local3DPoint pexit (r.Gaus(0,3), r.Gaus(0,3), r.Gaus(50,20));
      float pabs = 100*r.Rndm();
      float tof  = r.Gaus(1e-6,1e-8);
      float eloss= r.Landau(1e-3,1e-7);
      int ptype  = i%2;
      int detId  = i%20;
      int trackId= i%100;
      hit = new PSimHit(pentry,pexit,pabs,tof,eloss,ptype,detId,trackId);

      T.Fill();
   }

   T.Print();   //Print Tree statistics
   T.Write();   //Write Tree header to the file
}
```

Browsing the PSimHit Tree

split = 0



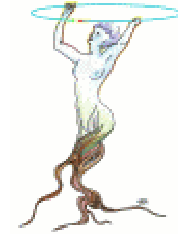



```
*Tree    :T          : CMS demo tree                                         *
*Entries :    50000 : Total =          4703775 bytes  File  Size =    2207143 *
*        :          : Tree compression factor =   2.13                        *
*******************************************************************************
*Br    0 :hit       :                                                         *
*Entries :    50000 : Total  Size=    4703775 bytes  File Size  =    2207143 *
*Baskets :      295 : Basket Size=      16000 bytes  Compression=   2.13     *
*.............................................................................*
```

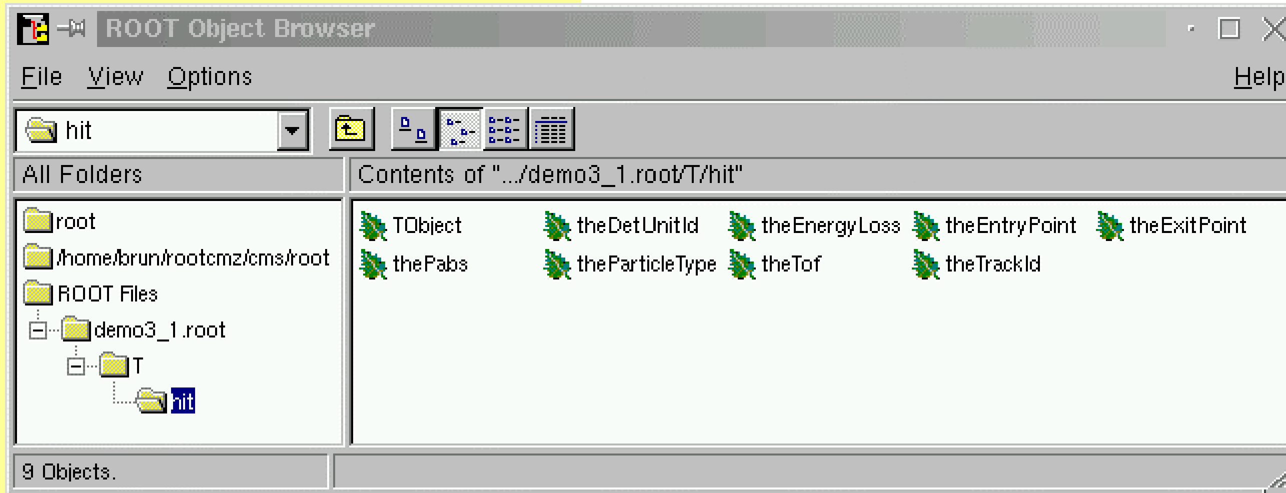
1 branch only

Browsing the PSimHit Tree

split = 1



```
*****************************************************************************
*Tree    :T         : CMS demo tree                                          *
*Entries :    50000 : Total =         5258415 bytes  File  Size =    2021907 *
*        :          : Tree compression factor =   2.60                       *
******************************************************************************
*Branch  :hit                                                                *
*Entries :    50000 : BranchElement (see below)                              *
*...................................
*Br    0 :TObject   :
*Entries :    50000 : Total  Size=
*Baskets :       56 : Basket Size=
*...................................
*Br    1 :theEntryPoint :
*Entries :    50000 : Total  Size=
*Baskets :      119 : Basket Size=
*...................................
*Br    2 :theExitPoint :
*Entries :    50000 : Total  Size=
*Baskets :      119 : Basket Size=
*...................................
*Br    3 :thePabs   :
*Entries :    50000 : Total  Size=
*Baskets :       12 : Basket Size=
*...................................
*Br    4 :theTof    :
*Entries :    50000 : Total  Size=
*Baskets :       12 : Basket Size=     16000 bytes  Compression=   1.32      *
*............................................................................*
*Br    5 :theEnergyLoss :                                                    *
*Entries :    50000 : Total  Size=     191964 bytes  File Size  =     122761 *
*Baskets :       12 : Basket Size=     16000 bytes  Compression=   1.56      *
*............................................................................*
*Br    6 :theParticleType :                                                  *
*Entries :    50000 : Total  Size=     191988 bytes  File Size  =       1860 *
*Baskets :       12 : Basket Size=     16000 bytes  Compression= 103.22      *
*............................................................................*
*Br    7 :theDetUnitId :                                                     *
*Entries :    50000 : Total  Size=     191952 bytes  File Size  =       2298 *
*Baskets :       12 : Basket Size=     16000 bytes  Compression=  83.53      *
*............................................................................*
*Br    8 :theTrackId :                                                       *
*Entries :    50000 : Total  Size=     191976 bytes  File Size  =       4179 *
*Baskets :       12 : Basket Size=     16000 bytes  Compression=  45.94      *
*............................................................................*
```

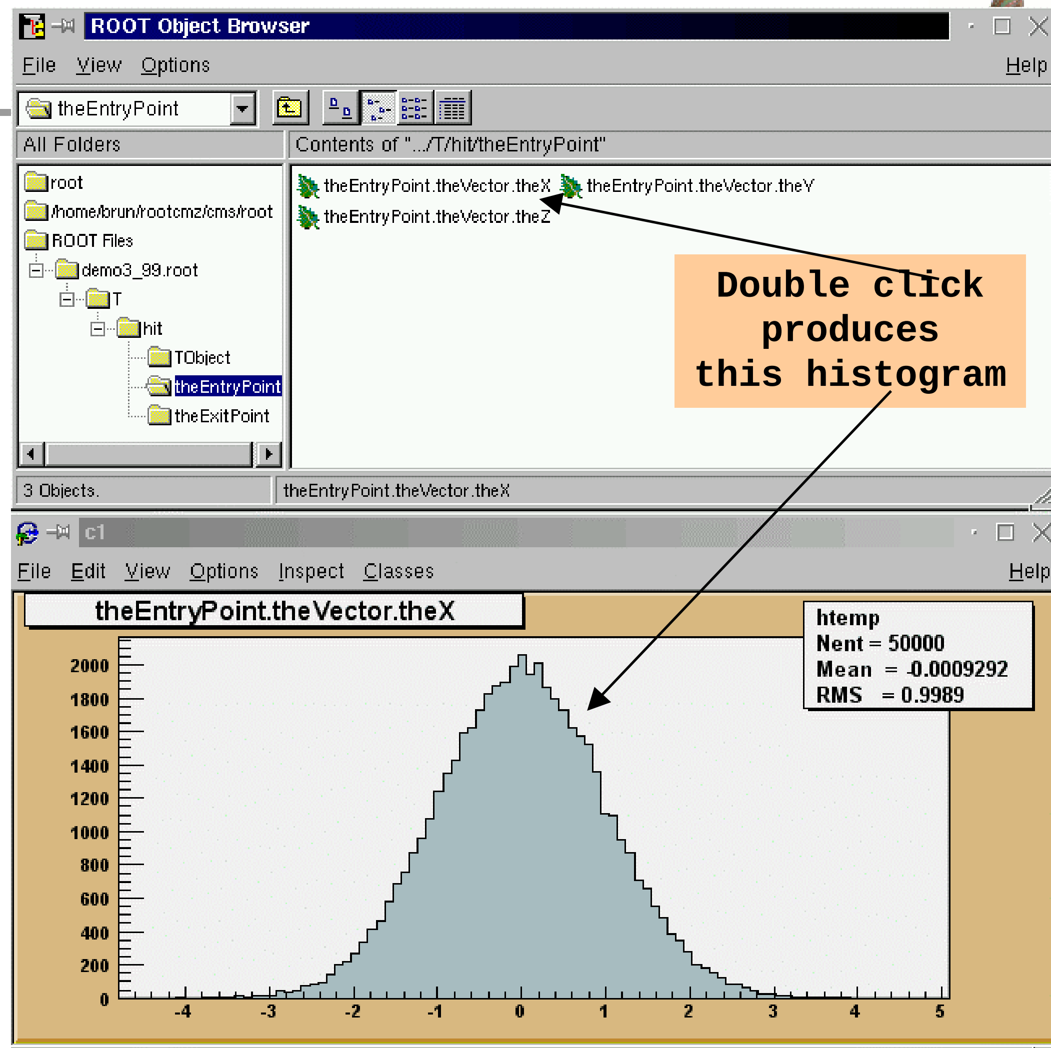


9 branches

Browsing the PSimHit Tree

split = 99

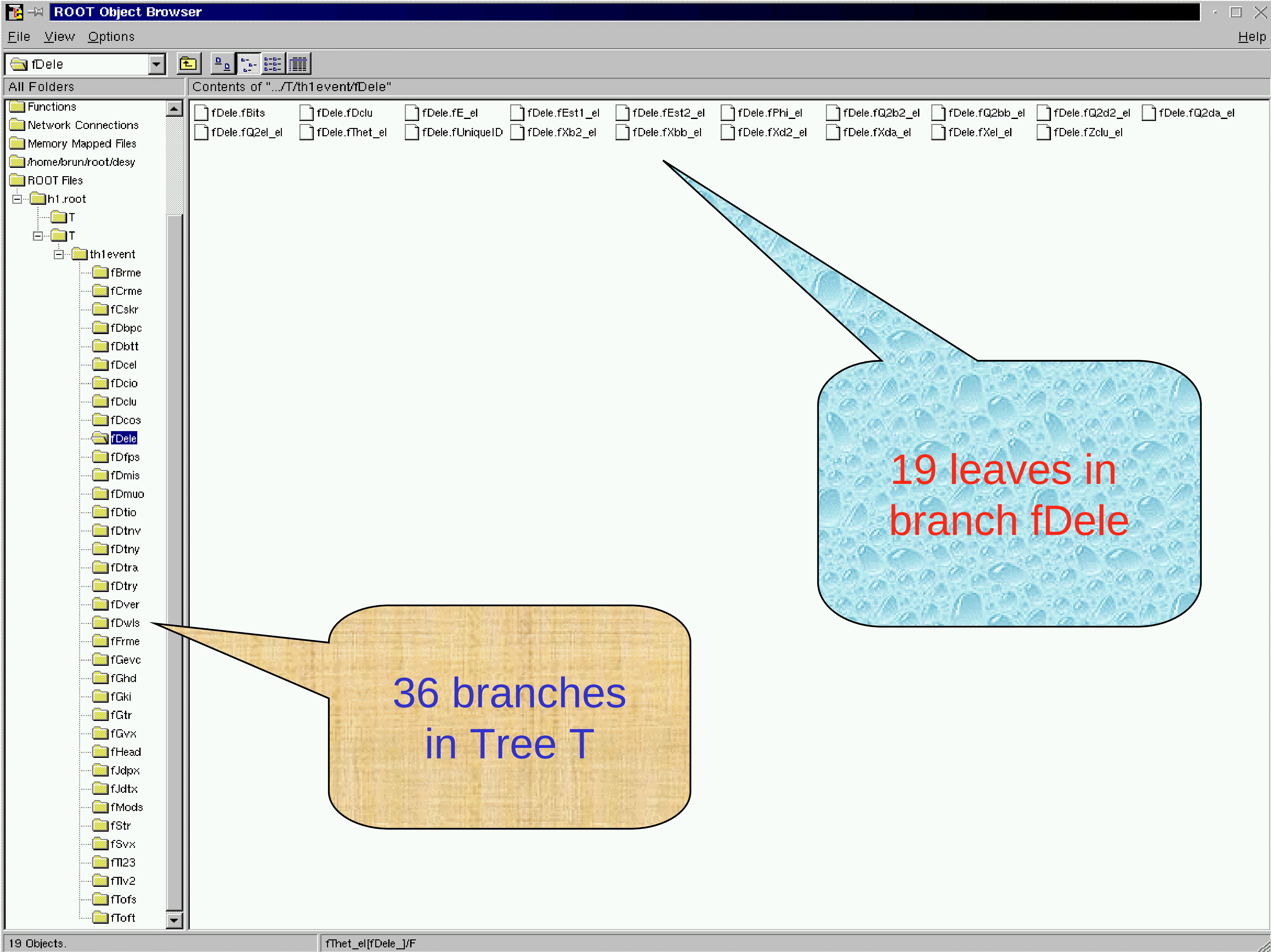
```
*******************************************************************************
*Tree    :T         : CMS demo tree                                          *
*Entries :    50000 : Total =         2687592 bytes  File  Size =    1509041 *
*        :          : Tree compression factor =   1.78                       *
*******************************************************************************
*Branch  :hit                                                                *
*Entries :    50000 : BranchElement (see below)                              *
*............................................................................*
*Br    0 :fUniqueID :                                                        *
*Entries :    50000 : Total  Size=     191964 bytes  File Size  =       1272 *
*Baskets :       12 : Basket Size=      16000 bytes  Compression= 150.92     *
*............................................................................*
*Br    1 :fBits     :                                                        *
*Entries :    50000 : Total  Size=     191964 bytes  File Size  =       1260 *
*Baskets :       12 : Basket Size=      16000 bytes  Compression= 152.35     *
*............................................................................*
*Br    2 :theEntryPoint :                                                    *
*Entries :    50000 : Total  Size=          0 bytes  File Size  =          0 *
*Baskets :        0 : Basket Size=      16000 bytes  Compression=   1.00     *
*............................................................................*
*Br    3 :theEntryPoint.theVector.theX :                                     *
*Entries :    50000 : Total  Size=     191952 bytes  File Size  =     177959 *
*Baskets :       12 : Basket Size=      16000 bytes  Compression=   1.08     *
*............................................................................*
*Br    4 :theEntryPoint.theVector.theY :                                     *
*Entries :    50000 : Total  Size=     191952 bytes  File Size  =     177934 *
*Baskets :       12 : Basket Size=      16000 bytes  Compression=   1.08     *
*............................................................................*
*Br    5 :theEntryPoint.theVector.theZ :                                     *
*Entries :    50000 : Total  Size=     191952 bytes  File Size  =     178312 *
*Baskets :       12 : Basket Size=      16000 bytes  Compression=   1.08     *
*............................................................................*
*Br    6 :theExitPoint :                                                     *
*Entries :    50000 : Total  Size=          0 bytes  File Size  =          0 *
*Baskets :        0 : Basket Size=      16000 bytes  Compression=   1.00     *
*............................................................................*
*Br    7 :theExitPoint.theVector.theX :                                      *
*Entries :    50000 : Total  Size=     191988 bytes  File Size  =     178060 *
*Baskets :       12 : Basket Size=      16000 bytes  Compression=   1.08     *
*............................................................................*
*Br    8 :theExitPoint.theVector.theY :                                      *
*Entries :    50000 : Total  Size=     191988 bytes  File Size  =     178072 *
*Baskets :       12 : Basket Size=      16000 bytes  Compression=   1.08     *
*............................................................................*
*Br    9 :theExitPoint.theVector.theZ :                                      *
*Entries :    50000 : Total  Size=     191988 bytes  File Size  =     168655 *
*Baskets :       12 : Basket Size=      16000 bytes  Compression=   1.14     *
*............................................................................*
*Br   10 :thePabs   :                                                        *
*Entries :    50000 : Total  Size=     191988 bytes  File Size  =     170871 *
*Baskets :       12 : Basket Size=      16000 bytes  Compression=   1.12     *
*............................................................................*
*Br   11 :theTof    :                                                        *
*Entries :    50000 : Total  Size=     191976 bytes  File Size  =     145548 *
*Baskets :       12 : Basket Size=      16000 bytes  Compression=   1.32     *
*............................................................................*
*Br   12 :theEnergyLoss :                                                    *
*Entries :    50000 : Total  Size=     191964 bytes  File Size  =     122761 *
*Baskets :       12 : Basket Size=      16000 bytes  Compression=   1.56     *
*............................................................................*
*Br   13 :theParticleType :                                                  *
*Entries :    50000 : Total  Size=     191988 bytes  File Size  =       1860 *
*Baskets :       12 : Basket Size=      16000 bytes  Compression= 103.22     *
*............................................................................*
*Br   14 :theDetUnitId :                                                     *
*Entries :    50000 : Total  Size=     191952 bytes  File Size  =       2298 *
*Baskets :       12 : Basket Size=      16000 bytes  Compression=  83.53     *

*............................................................................*
*Br   15 :theTrackId :                                                       *
*Entries :    50000 : Total  Size=     191976 bytes  File Size  =       4179 *
*Baskets :       12 : Basket Size=      16000 bytes  Compression=  45.94     *
*............................................................................*
```

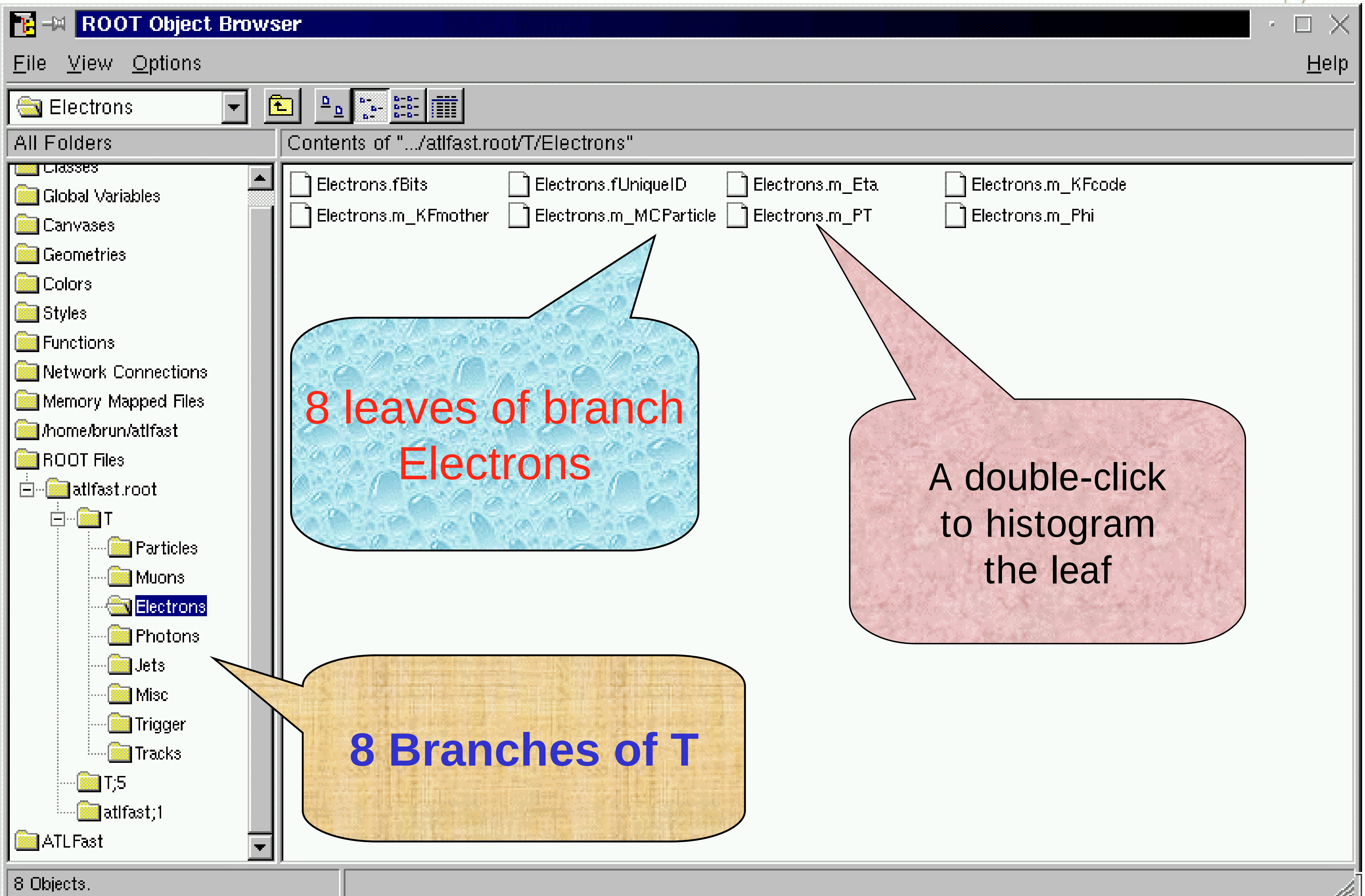


16 branches

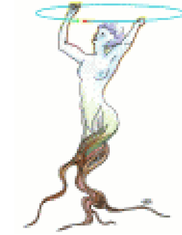
Collections of Hits

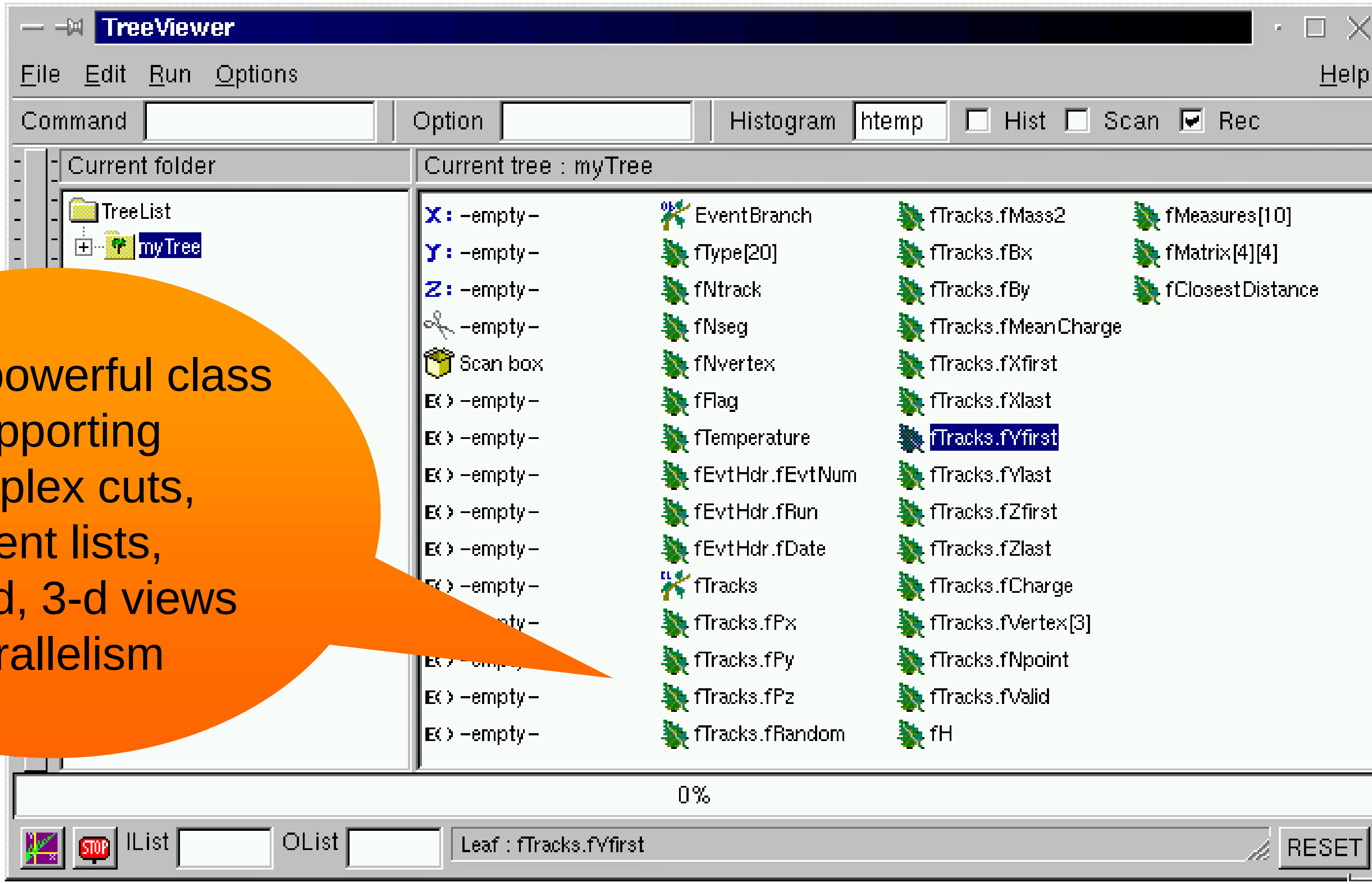
- A more realistic Tree will have
  - A collection of Detectors
  - Each detector one or more collection of hits


ROOT Object Browser
File View Options Help
fDele
All Folders
Contents of ".../T/th1event/fDele"
Functions
Network Connections
Memory Mapped Files
/home/brun/root/desy
ROOT Files
h1.root
T
T
th1event
fBrme
fCrme
fCskr
fDbpc
fDbtt
fDcel
fDcio
fDclu
fDcos
fDele
fDfps
fDmis
fDmuo
fDtio
fDtnv
fDtny
fDtra
fDtry
fDver
fDwls
fFrme
fGevc
fGhd
fGki
fGtr
fGvx
fHead
fJdpx
fJdtx
fMods
fStr
fSvx
fTl23
fTlv2
fTofs
fToft
fDele.fBits
fDele.fDclu
fDele.fE_el
fDele.fEst1_el
fDele.fEst2_el
fDele.fPhi_el
fDele.fQ2b2_el
fDele.fQ2bb_el
fDele.fQ2d2_el
fDele.fQ2da_el
fDele.fQ2el_el
fDele.fThet_el
fDele.fUniqueID
fDele.fXb2_el
fDele.fXbb_el
fDele.fXd2_el
fDele.fXda_el
fDele.fXel_el
fDele.fZclu_el
19 leaves in branch fDele
36 branches in Tree T
19 Objects.
fThet_el[fDele_]/F


ROOT Object Browser
File View Options Help
Electrons
All Folders
Contents of ".../atlfast.root/T/Electrons"
Global Variables
Canvases
Geometries
Colors
Styles
Functions
Network Connections
Memory Mapped Files
/home/brun/atlfast
ROOT Files
atlfast.root
T
Particles
Muons
Electrons
Photons
Jets
Misc
Trigger
Tracks
T;5
atlfast;1
ATLFast
Electrons.fBits
Electrons.fUniqueID
Electrons.m_Eta
Electrons.m_KFcode
Electrons.m_KFmother
Electrons.m_MCParticle
Electrons.m_PT
Electrons.m_Phi
8 Objects.
8 leaves of branch Electrons
A double-click to histogram the leaf
8 Branches of T

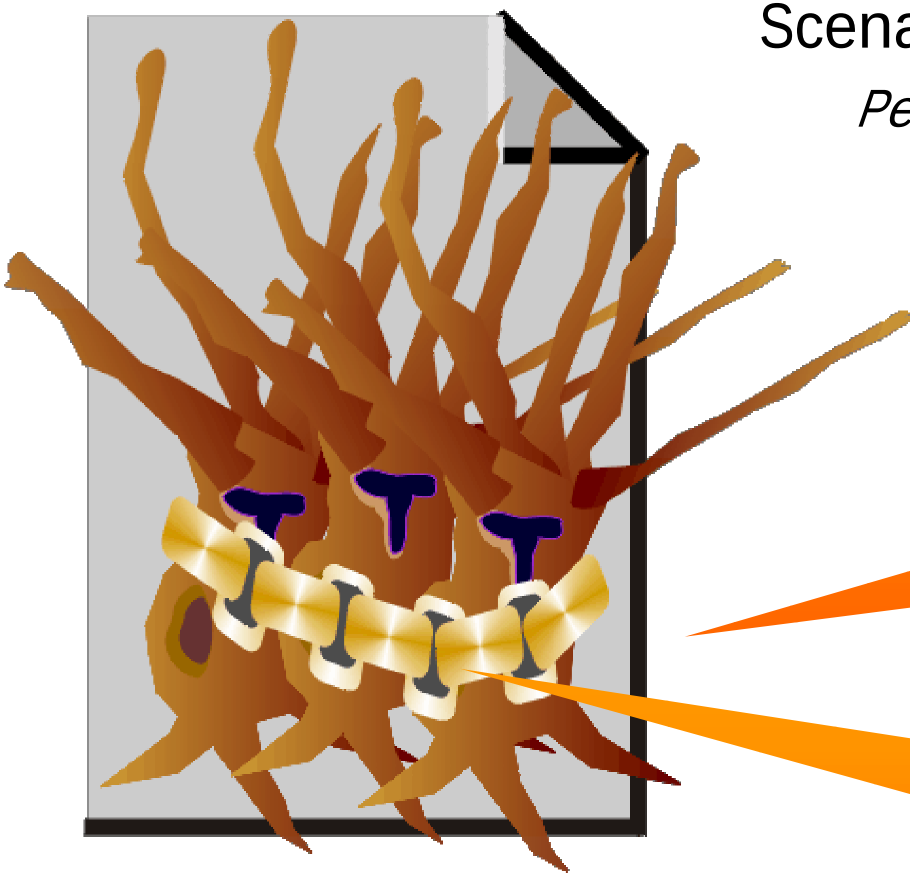
The Tree Viewer & Analyzer





A very powerful class
supporting
complex cuts,
event lists,
1-d,2-d, 3-d views
parallelism

Chains



Scenario:

Perform an analysis using multiple ROOT files. All files are of the same structure and have the same tree.

Chains are collections of chains or files

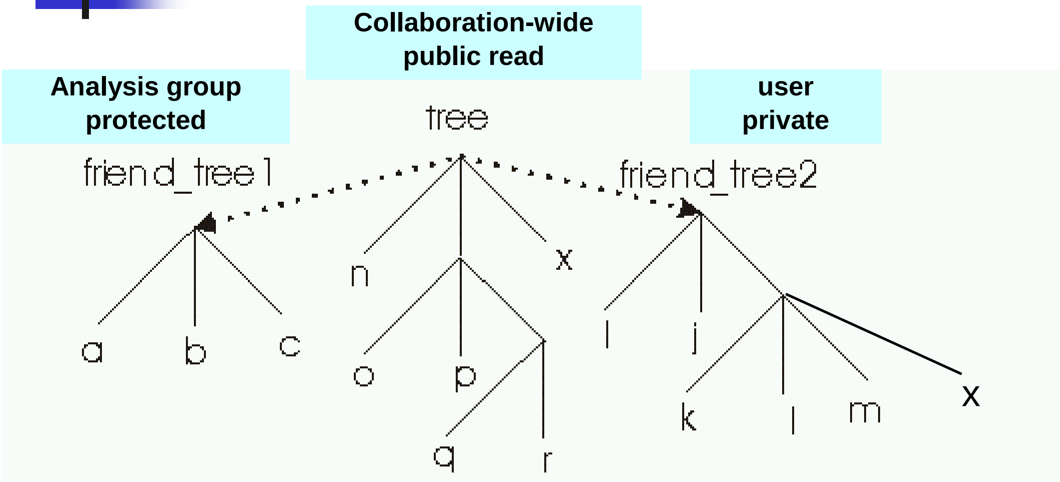
Chains can be built automatically by quering the run/file catalog

Chains of Trees

- A TChain is a collection of Trees.
- Same semantics for TChains and TTrees
  - root > .x h1chain.C
  - root > chain.Process(“h1analysis.C”)

```
{
  //creates a TChain to be used by the h1analysis.C class
  //the symbol H1 must point to a directory where the H1 data sets
  //have been installed

  TChain chain("h42");
    chain.Add("$H1/dstarmb.root");
    chain.Add("$H1/dstarp1a.root");
    chain.Add("$H1/dstarp1b.root");
    chain.Add("$H1/dstarp2.root");
}
```

Tree Friends



Processing time independent of the number of friends unlike table joins in RDBMS

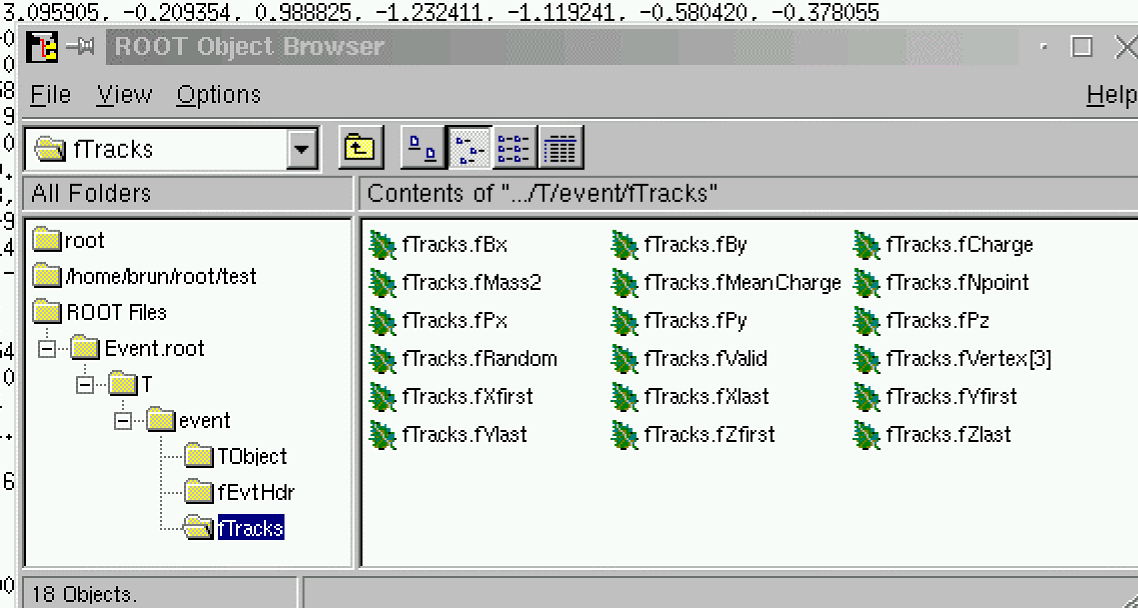
```
Root > TFile f1(“tree1.root”);
Root > tree.AddFriend(“tree2”,“tree2.root”)
Root > tree.AddFriend(“tree3”,“tree3.root”);
Root > tree.Draw(“x:a”,”k<c”);
Root > tree.Draw(“x:tree2.x”,”sqrt(p)<b”);
```

The “No Shared Library” case

- There are many applications for which it does not make sense to read data without the code of the corresponding classes.
- In true OO, you want to exploit Data Hiding and rely on the functional interface.
- However, there are also cases where the functional interface is not necessary (PAW ntuples).
- It is nice to be able to browse any type of file without any code. May be you cannot do much, but it gives some confidence that you can always read your data sets.
- We have seen a religious debate on this subject.
- Our conclusion was that we had to support these two modes of operation.
- Support for the “No Shared Lib case” is non trivial

read/query Trees without the classes

```
root [0] TFile f("Event.root")
Warning in <TClass::TClass>: no dictionary for class Event is available
Warning in <TClass::TClass>: no dictionary for class EventHeader is available
Warning in <TClass::TClass>: no dictionary for class Track is available
root [1] T.Show(45)
======> EVENT:45
 fUniqueID       = 0
 fBits           = 50331648
 fType[20]       = 116 121 112 101 48 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 fNtrack         = 609
 fNseg           = 6066
 fNvertex        = 7
 fFlag           = 1
 fTemperature    = 20.529432
 fEvtHdr.fEvtNum = 45
 fEvtHdr.fRun    = 200
 fEvtHdr.fDate   = 960312
 fTracks         = 609
 fTracks.fPx     = -0.077692, -2.625842, 1.130895, 3.095905, -0.209354, 0.988825, -1.232411, -1.119241, -0.580420, -0.378055
 fTracks.fPy     = 0.332117, 0.482258, -0.789722, -0
 fTracks.fPz     = 0.341083, 2.669760, 1.379341, 3.0
 fTracks.fRandom = 529.431580, 529.431580, 529.43158
 fTracks.fMass2  = 8.900000, 8.900000, 8.900000, 8.9
 fTracks.fBx     = -0.042621, 0.069631, -0.141984, 0
 fTracks.fBy     = 0.001728, 0.116303, 0.046655, -0.
 fTracks.fMeanCharge = 0.001209, 0.001738, 0.006828,
 fTracks.fXfirst = 2.092409, 0.549266, -1.027804, -9
 fTracks.fXlast  = 5.749741, 15.938519, 0.260784, 14
 fTracks.fYfirst = 10.398095, 2.455857, 16.579025, -
 fTracks.fYlast  = -7.106707, 13.617641, 12.946399,
 fTracks.fZfirst = 56.846382, 39.277451, 47.035370,
 fTracks.fZlast  = 211.758606, 213.753479, 217.62054
 fTracks.fCharge = 0.000000, 0.000000, 1.000000, 0.0
 fTracks.fVertex[3] = -0.181014 0.105711 -20.070364
0.034267 0.096083 -3.769124 , -0.119004 0.038680 -1.
79879 , -0.165064 0.044276 -9.960069
 fTracks.fNpoint = 64, 60, 66, 61, 62, 61, 62, 64, 6
 fTracks.fValid  = 1, 0, 1, 0, 1, 1, 0, 1, 0, 1
 fH              = (TH1F*)8a93ed0
 fMeasures[10]   = -2 0 5 2 5 1 14 13 9 11
 fMatrix[4][4]   = -0.625079 0.530725 -0.786688 0.00
.000000 0.000000
 fClosestDistance = 1.441706 1.708780 -0.655489 1.539576 0.804130 0.048241 -0.594909
root [2] new TBrowser
(class TBrowser*)0x88b7460
root [3]
```



TFile::MakeProject

Generate the classes header files
Compile them
make a shared lib
link the shared lib

```
root [4] f.MakeProject("xx","*","recreate++")
MakeProject has generated 21 classes in xx
xx/MAKE file has been generated
Shared lib xx/xx.so has been generated
Shared lib xx/xx.so has been dynamically linked
root [5] ATLFElectron e
root [6] e.Dump()
m_KFcode                     11          Electron KF-code
m_MCParticle                 6           Electron position in MCParticles list
m_KFmother                   0           Electron mother KF-code
m_Eta                        0           Eta coordinate
m_Phi                        0           Phi coordinate
m_PT                         0           Transverse energy
fUniqueID                    0           object unique identifier
fBits                        50331648    bit field status word
```

TFile::MakeProject

```
////////////////////////////////////////////////////////////
//   This class has been generated by TFile::MakeProject
//     (Mon May 28 19:34:37 2001 by ROOT version 3.01/03)
//      from the StreamerInfo in file atlfast.root
////////////////////////////////////////////////////////////


#ifndef ATLFElectron_h
#define ATLFElectron_h

#include "TObject.h"
#include "TAtt3D.h"

class ATLFElectron : public TObject , public TAtt3D {

public:
   Int_t        m_KFcode;        //Electron KF-code
   Int_t        m_MCParticle;    //Electron position in MCParticles l
   Int_t        m_KFmother;      //Electron mother KF-code
   Float_t      m_Eta;           //Eta coordinate
   Float_t      m_Phi;           //Phi coordinate
   Float_t      m_PT;            //Transverse energy

   ATLFElectron() {;}
   virtual ~ATLFElectron() {;}

   ClassDef(ATLFElectron,1) //
};

   ClassImp(ATLFElectron)
#endif
```

All necessary header files are included

Comments preserved

Can do I/O Inspect Browse,etc

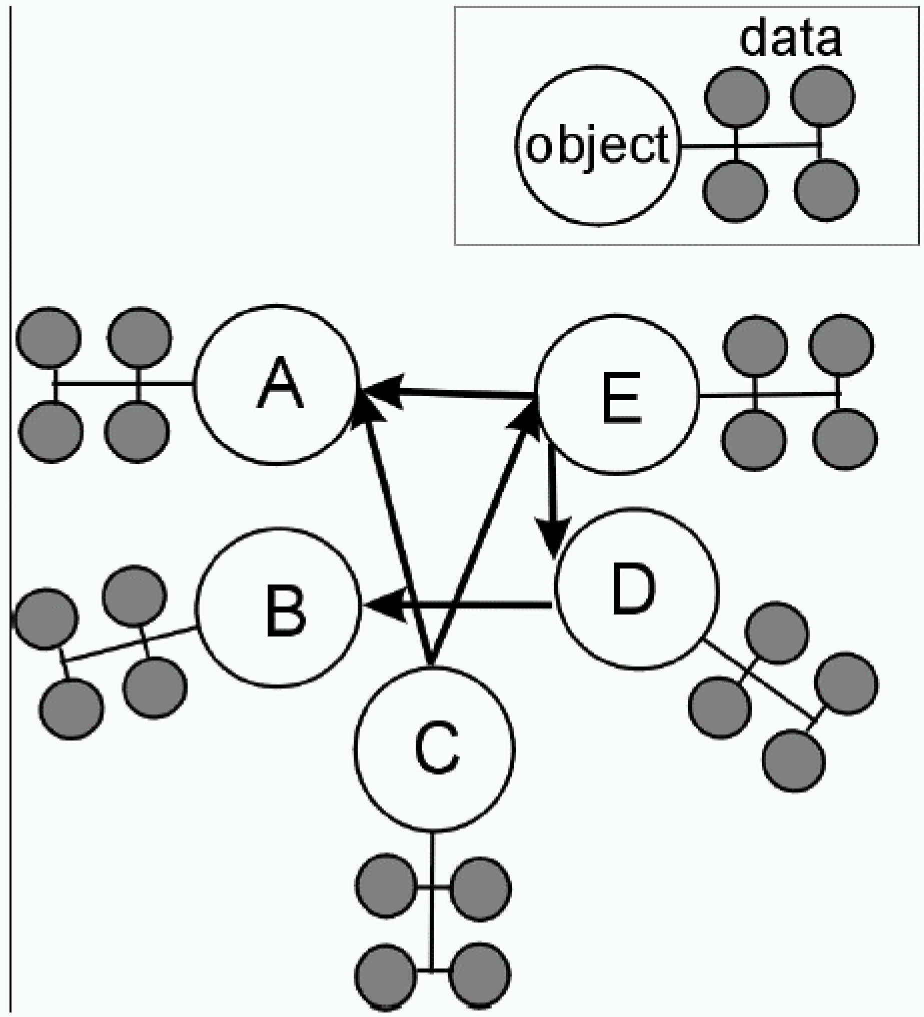
ROOT Folders

Why Folders ?

This diagram shows a system without folders. The objects have pointers to each other to access each other's data.

Pointers are an efficient way to share data between classes. However, a direct pointer creates a direct coupling between classes.

This design can become a very tangled web of dependencies in a system with a large number of classes.

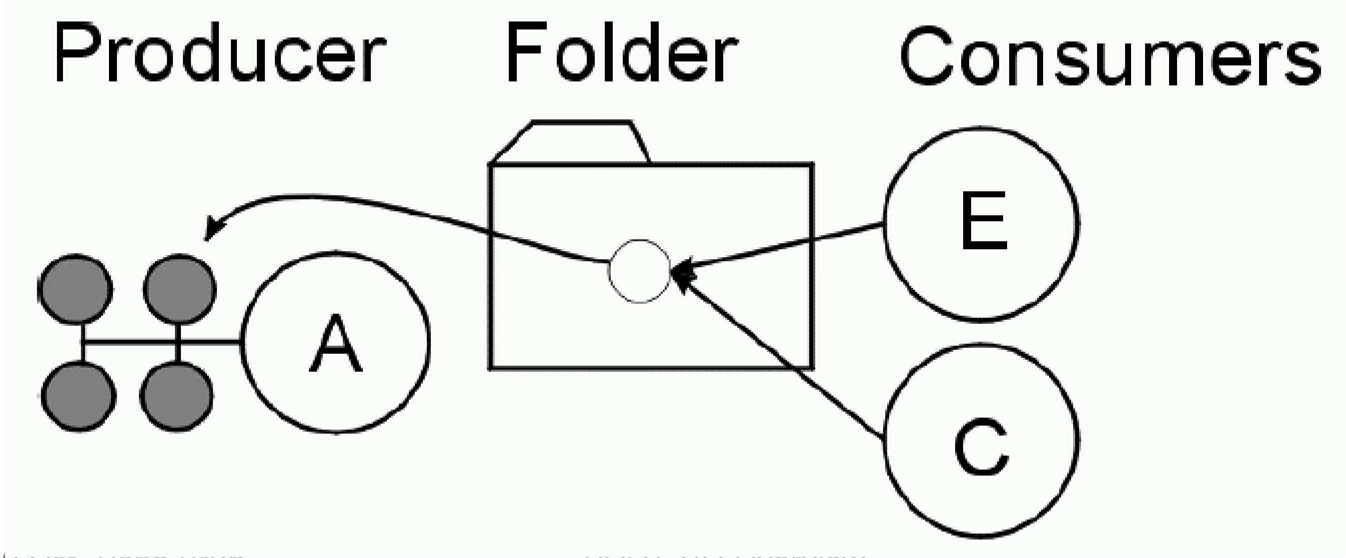


Why Folders ?

In the diagram below, a reference to the data is in the folder and the consumers refer to the folder rather than each other to access the data.

The naming and search service provided by the ROOT folders hierarchy provides an alternative. It loosely couples the classes and greatly enhances I/O operations.

In this way, folders separate the data from the algorithms and greatly improve the modularity of an application by minimizing the class dependencies.



Posting Data to a Folder
(Producer)

- No changes required in user class structure.
- Build a folder structure with:
  - TFolder::AddFolder(TFolder *)
- Post objects or collections to a Folder with:
  - TFolder::Add(TObject*)
- A TFolder can contain other folders or any TObject descendents. In general, users will not post a single object to a folder, they will store a collection or multiple collections in a folder. For example, to add an array to a folder:
  - TObjArray *array;
  - run_mc->Add(array);

Reading Data from a Folder

(Consumer)

One can search for a folder or an object in a folder using the TROOT::FindObjectAny method. FindObjectAny analyzes the string passed as its argument and searches in the hierarchy until it finds an object or folder matching the name.
With FindObjectAny, you can give the full path name, or the name of the folder.
If only the name of the folder is given, it will return the first instance of that name.

```
conf = (TFolder*)gROOT->FindObjectAny("/aliroot/Run/Configuration");
 or
conf = (TFolder*)gROOT->FindObjectAny("Configuration");
```

A string-based search is time consuming. If the retrieved object is used frequently or inside a loop, you should save a pointer to the object as a class data member.

Use the naming service only in the initialization of the consumer class.

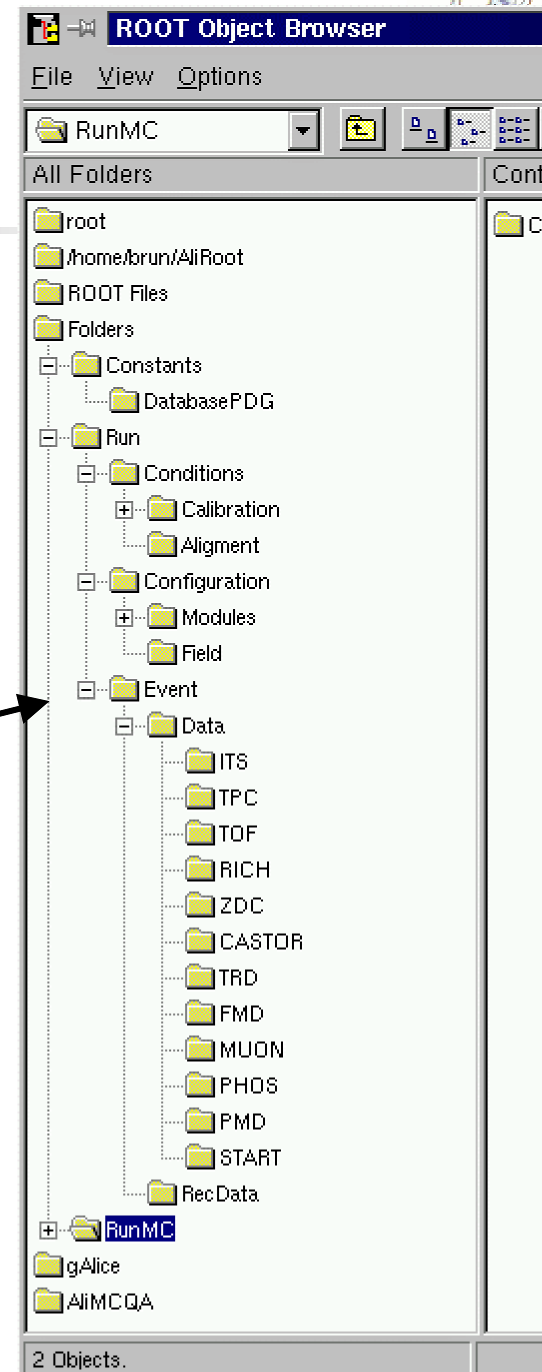
Example: Alice folders

Some of the AliRoot folders shown in the browser:

A ROOT Tree can be automatically generated from the folder, eg:

TTree T("T","/Event");

T.Fill();



ROOT working with Objectivity

http://www.phenix.bnl.gov/WWW/publish/onuchin/rooObjy/

ROOT
Objectivity

- Introduction
- Naming Rules
- Examples
- FAQ
- Class Reference
- Availability
- TODO
- PHENIX DBrowser

Introduction

This is library of ROOT wrappers around Objectivity base classes and functions.

- **The Goals:**
  - Create a tool library which will allow the retrieval of data from Objy DB from a ROOT prompt.
  - Provide the basis library for ROOT GUI PHENIX Database Browser
  - The library was developed with hope it can be used inside PHENIX software framework
- **The Status:**
  - More than 40 Objectivity classes were wrapped with 10k lines of code and filled with Objy docs. Including:
    - Objectivity object handler and iterator classes, i.e. ooHandle(XXX), ooItr(XXX)
    - Objectivity Active Schema classes
    - Objectivity global functions, constants etc.
  - Compiled on Linux. Tested ++ more testing required. Not ported to S
- **The library allows users to:**
  - connect to Federated database
  - open up database
  - iterate through databases/containers
  - obtain descriptions of the classes in database
  - retrieve persistent data for any object in the database
  - ++ much more, interactively from ROOT

If you have any comments about this page please send them to Valeriy Onuchin

Objy and ROOT
can work together
An interactive interface
developed by Phenix

ROOT working with Oracle

http://www.phenix.bnl.gov/WWW/publish/onuchin/RDBC/

See also: **http://www.gsi.de/computing/root/OracleAccess.htm**

ROOT
ORACLE
mSQL
PostgreSQL

- Introduction
- Demos & Tests
- Class Reference
- Download & Install

Introduction

- This library is a set of ROOT wrappers of libodbc++ classes that provides an interface based on the JDBC API.
- It corresponds to JDBC–ODBC bridge in Java terminology and makes it possible to do:
  1. establish a connection from ROOT sesssion to any database for which ODBC driver available.
  2. send SQL statements.
  3. process the results.
- Major differences between libodbc++ and RDBC:
  - RDBC written according to ROOT coding conventions. All method names begin with upper case letter.
  - Since ROOT C++ interpreter (CINT) does not provide full support of namespaces, all RDBC classes have TSQL prefix, e.g. TSQLStatement corresponds to odbc::Statement etc..
  - Since ROOT C++ interpreter (CINT) does not provide full support of exceptions, **Rt** object communication mechanism is used for *exception handling* (see TSQL::SetHandler method).
  - RDBC supprots
    - TSQLResultSet::GetObject
    - TSQLResultSet::UpdateObject
    - TSQLPreparedStatement::SetObject

    methods which allow to write/read ROOT objects to/from database (see "Save Your Histos in Oracle Database!" example).
- We consider RDBC as a low–level API and a base for higher–level interface between ROOT based offline enviroment of MINOS experiment and Oracle database. New ROOT–RDBC based GUI tools for Oracle are coming. Stay tuned!

ODBC compliant interface to Oracle

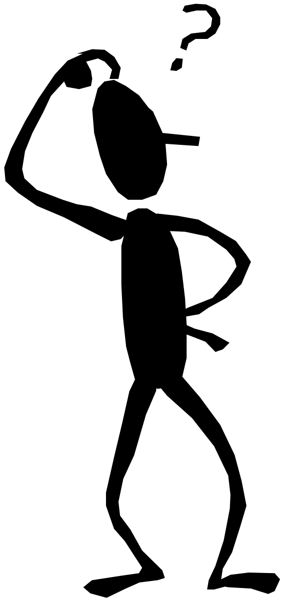
Time to conclude

We have a working system

Used by many people

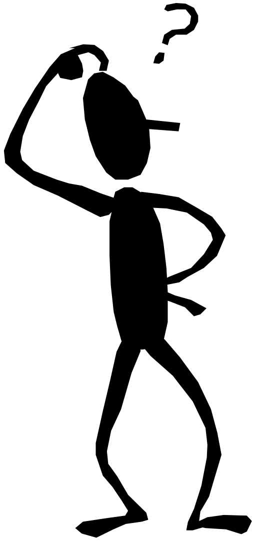
In many different configurations

but



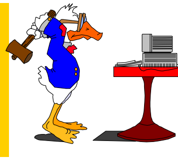
General remarks

- In 1995, we had planned less than 50% of ROOT 2001.
  - \- importance of dictionary, RTTI
  - \- Automatic Schema Evolution
  - \- effort in GUI
  - \- Online requirements (Threads, Timers, Sockets, etc)



- Development of a system is driven by:
  - \- ideas from authors
  - \- ideas from users
  - \- new ideas and techniques in computing
  - \- OS development. In 1995, push for Windows, Linux not here
  - \- language developments (eg template support, exception handling, Java)
  - \- cooperation with other systems (ex Objy, Oracle, Corba, Qt, etc)
  - \- manpower

Users expect stable and working systems. Quality of a system should improve with time. Often in contradiction with major developments.



ROOT: an Evolving System

- The ROOT system has been in continuous development since 1995 surviving major changes, major enhancements and an ever increasing number of users.
- In the same way that Root2001 is far from the original Root1995, we expect that Root2006 will include many contributions reflecting the continuous changes and new ideas in the field of computing.
- This implies a strong cooperation between software developers in the major experiments.
- Root is being developed in very close cooperation with a cloud of software developers in small, medium and large experiments. Computer scientists from non-HEP fields are also contributing.

Download source, Binaries

http://root.cern.ch

- Intel x86 Linux for Redhat 7.1 (glibc 2.2) and gcc 2.96, version 3.02/01 (8.1 MB). NEW
- Intel x86 Linux for Redhat 6.1 (glibc 2.1) and gcc2.95.2, version 3.02/01 (9.7 MB). NEW
- Intel x86 Linux for Redhat 6.1 (glibc 2.1) and egcs1.1.2, version 3.02/01 (8.6 MB). NEW
- Intel x86 Linux for Redhat 5.0/5.1/5.2 (glibc) and egcs 1.1.1, version 3.02/01 (8.6 MB). NEW
- Intel Itanium Linux for Redhat 7.0 (glibc 2.2) and gcc 2.96, version 3.00/06 (9.0 MB). NEW
- HP-UX 10.20 with aCC (v1.18), version 3.02/01 (12.7 MB). NEW
- Compaq Alpha OSF1 with cxx 6.2, version 3.02/01 (9.3 MB). NEW
- Compaq Alpha OSF1 with egcs 1.1.2, version 3.02/01 (10.6 MB). NEW
- Compaq Alpha Linux with egcs 1.1.2, version 3.01/06 (11.0 MB). NEW
- Compaq iPAQ PocketPC Linux with gcc 2.95, version 3.00/06 (6.7 MB). NEW
  For more on Linux on iPAQ see www.handhelds.org.
- AIX 4.3 with xlC version 3, version 3.02/01 (11.3 MB, works only on AIX 4.3). NEW
- AIX 4.5 with xlC version 5, version 3.02/01 (11.3 MB, works only on AIX 4.5). NEW
- Sun SPARC Solaris 5.6 with CC4.2, version 3.02/01 (8.5 MB). NEW
  It cannot be used with Solaris 5.7 or 5.8 even using the same compiler version. You must recompile from the source on these two systems.
- Sun SPARC Solaris 5.7 with CC5.0, version 3.02/01 (9.7 MB). NEW
  It cannot be used with Solaris 5.6 or 5.8 even using the same compiler version. You must recompile from the source on these two systems.
- Sun SPARC Solaris 5.8 with CC5.2, version 3.02/01 (10.1 MB). NEW
  It cannot be used with Solaris 5.6 or 5.7 even using the same compiler version. You must recompile from the source on these two systems.
- SGI IRIX 6.5 with CC, version 3.02/01 (compiled with -n32) (10.5 MB). NEW
- SGI IRIX 6.5 with g++ 2.95.2, version 3.02/01 (11.6 MB). NEW
- SGI IRIX 6.5 with KCC, version 3.02/01 (10.1 MB). NEW
- LinuxPPC/2000 (glibc 2.1) gcc 2.95, version 3.01/05 (7.8 MB). NEW
  Thanks to Damir Buskulic (buskulic@lapp.in2p3.fr) for building this version.
- Windows/NT/95/98 with VC++ 6.0, version 3.02/01 (good old tar file) (9.3 MB). NEW
- Windows/NT/95/98 with VC++ 6.0,compiled with debug info, version 3.02/01 (good old tar file) (17.4 MB). NEW
- Windows/NT/95/98 with VC++ 6.0, version 3.02/00 (built with InstallShield) (9.1 MB). NEW
  When running from the MSDOS prompt, you must set the following environment variables, eg in your autoexec.bat: (Restart the system if you set these variables for the first time).

22 binary
tar balls
+ source

Makefiles

- 3 major OS (Unix, Windows, Mac OS/X)
- 10 different compilers
  - gcc with many flavors on nearly all platforms,
  - Solaris:CC4,5, HPUX:CC:aCC, SGI:CC, AIX:xlC
  - Alpha:CXX6, Windows:VC++6
  - KAI on SGI, Linux, Solaris
- 37 Makefiles

```
(pcnotebrun) [732] ls ~/root/config
ARCHS               Makefile.freebsd4        Makefile.linuxdeb2     Makefile.linuxsuse6  Makefile.solarisCC5
CVS                 Makefile.hpux            Makefile.linuxdeb2ppc  Makefile.lynxos      Makefile.solarisegcs
Makefile.aix        Makefile.hpuxacc         Makefile.linuxegcs     Makefile.macosx      Makefile.solarisgcc
Makefile.aixegcs    Makefile.hpuxegcs        Makefile.linuxia64gcc  Makefile.mklinux     Makefile.solariskcc
Makefile.alphacxx6  Makefile.in              Makefile.linuxia64sgi  Makefile.sgicc       Makefile.win32
Makefile.alphaegcs  Makefile.linux           Makefile.linuxkcc      Makefile.sgiegcs     config.in
Makefile.alphakcc   Makefile.linuxalphaegcs  Makefile.linuxpgcc     Makefile.sgikcc      root-config.in
Makefile.config     Makefile.linuxarm        Makefile.linuxppcegcs  Makefile.sgin32egcs  rootrc.in
Makefile.freebsd    Makefile.linuxdeb        Makefile.linuxrh42     Makefile.solaris
```

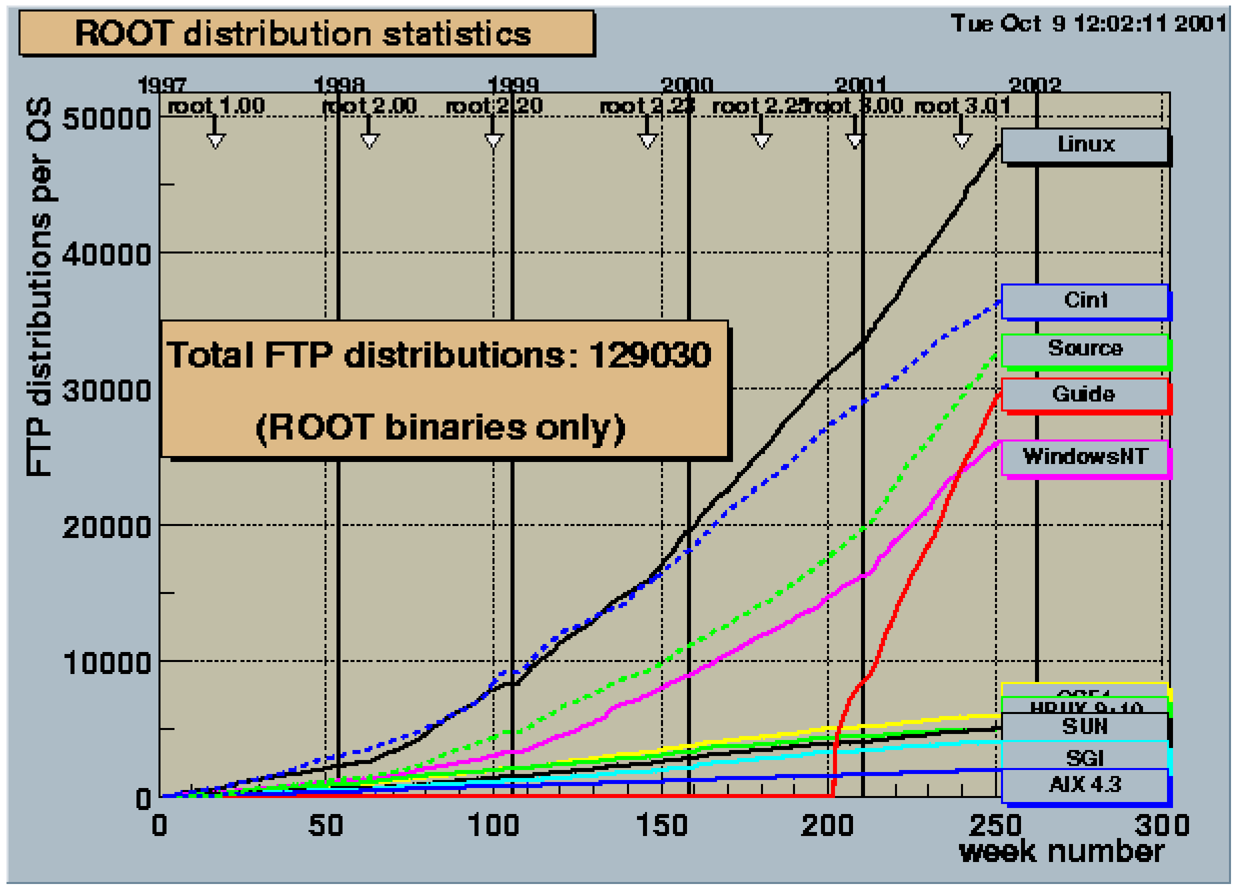
ROOT Downloads

129,000 binaries download

650,000 clicks per month

30,000 docs in 12 months

2200 reg users in roottalk

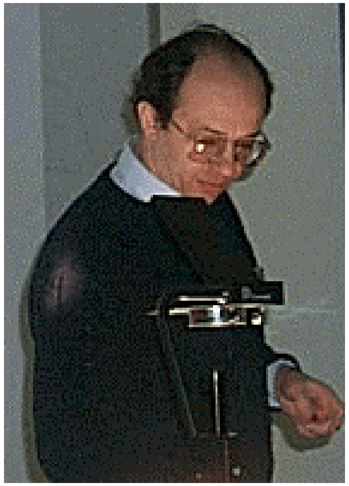


ROOT Users in the large experiments

ATLAS	133	WA98	25
ALICE	120	BELLE	22
CDF	88	COMPASS	22
PHENIX	86	KLOE	19
CMS	85	ALEPH	18
STAR	82	OPAL	17
JLAB	77	AUGER	16
D0	70	MINOS	16
BABAR	69	NOMAD	16
H1	48	BRAHMS	15
L3	43	GLAST	14
HERAB	37	AMS	12
NA49	37	NA45	12
LHCB	35	NA48	11
DELPHI	34	AMANDA	10
ZEUS	32	....	
HADES	27	....	
PHOBOS	27		

Registered users
in the ROOT system

ROOT team today



ROOT Educational Resources at FNAL

http://www-pat.fnal.gov/root/

Day 1: slides, Power Point file

- Overview of the ROOT Framework
- GUI basics
- Command line basics
- Finding Information

Day 2: slides, Power Point file

- Command Line (CINT)
- Scripts (CINT & ACLiC)
- Getting started with the
- Exercises Functions and Fitting
- TreeViewer

Day 3: slides, Power Point file

- Building Root Trees
- Reading Root Trees
- Using Trees in Analysis
- Add your class to ROOT

Examples: download all the example files used in the tutorial

Exercises: download the exercises we use in class, they are the same as we used at CERN school. Here is a list of the things that you will learn:

- three ways you can use ROOT: the command line, the script processor, and the graphical user interface (GUI).
- how to generate a PostScript file.
- histograms, and the input/output capabilities.
- example of an analysis using real physics data.
- example of a simulations and an event display.

C++ Basics for ROOT users
Follow this link for a refresher on C++.

ROOT User's Guide
Follow this link to the preliminary version of the ROOT User's Guide.
Hard copies of the ROOT User's Guide are now available in the Fermilab Stock Room.
The part number is: 1307–058000 and the cost is $5.21 a copy.

ROOT Mailing Lists
Follow this link to find the mailing lists and archives for roottalk and about root. This is a place where you can ask questions, and get an answer quickly. You can also scan the archive to find topics of interest.

Fermilab's ROOT Releases
Follow this link to see a list of ROOT releases available at Fermilab.