ROOT Progress Report

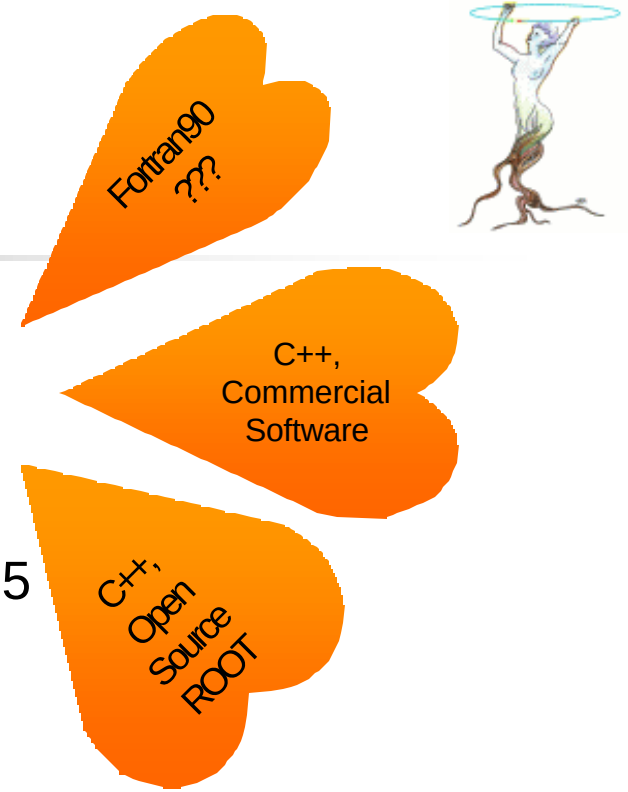
ROOT Workshop 2001 FNAL

June 13

René Brun

http://root.cern.ch

Project History



In 1994, fundamental divergence of opinions in Application Software group in IT. The PAW/Geant3 team is dismantled.

- Jan 95: Thinking/writing/rewriting/???
- November 95: Public seminar, show Root 0.5
- Spring 96: decision to use CINT
- Jan 97: Root version 1.0
- Jan 98: Root version 2.0
- Mar 99: Root version 2.21/08 (1st Root workshop FNAL)
- Feb 00: Root version 2.23/12 (2nd Root workshop CERN)
- Sep 00: Root version 2.25/03
- Mar 01: Root version 3.00/06
- Jun 01: Root version 3.01/05 (3rd Root workshop at FNAL)

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.

ROOT team today

Plan of talk

- In this talk, I will review the main developments in version 2.25, 3.00 and 3.01.
  - Focusing on the I/O aspects
- CINT will be covered by Masa
- System classes and GUI by Fons
- ROOT GUI on Windows by Fons and Bertrand
- Trees and related things: see my next talk
- Documentation: see Suzanne's talk

Version 2.25 (1) Sep 00

- In March 2000, move to CVS
  - Makefile system
  - Modularization of the libraries
- ACLIC
- TLatex, TSpline
- TPrincipal, TMultiDimFit
- TTree::MakeClass, MakeSelector
  - see talk on Trees
- TFolder, TTask
  - see talk on Folders and tasks
- Float_t --> Double_t in graphics classes + TH1 + TGraph

ACLIC

by Philippe Canal

- Automatic Compiler Linker Interface to CINT
- .x hsimple.cxx (via CINT)
- .x hsimple.cxx+ (via native compiler/linker)
  - second call to .x hsimple.cxx+ will not recompile if hsimple.cxx is not modified since previous call
- .x hsimple.cxx++ (same, force compilation)
- Same behavior on all systems
- We recommend to write always valid C++ with the include files specified. This works for C++ and CINT.

ACLIC example

Add includes normal C++

```
#include "TFile.h"
#include "TH1.h"
#include "TH2.h"
#include "TProfile.h"
#include "TNtuple.h"
#include "TRandom.h"

int hsimple()
{
  // Create a new ROOT binary machine independent file.
  // Note that this file may contain any kind of ROOT objects, histograms,
  // pictures, graphics objects, detector geometries, tracks, events, etc..
  // This file is now becoming the current directory.
  TFile hfile("hsimple.root","RECREATE","Demo ROOT file with histograms");

  // Create some histograms, a profile histogram and an ntuple
  TH1F *hpx    = new TH1F("hpx","This is the px distribution",100,-4,4);
  TH2F *hpxpy = new TH2F("hpxpy","py vs px",40,-4,4,40,-4,4);
  TProfile *hprof = new TProfile("hprof","Profile of pz versus px",100,-4,4,0,20);
  TNtuple *ntuple = new TNtuple("ntuple","Demo ntuple","px:py:pz:random:i");

  // Fill histograms randomly
  Float_t px, py, pz;
  for ( Int_t i=0; i<10000; i++) {
     gRandom->Rannor(px,py); //px and py will be two gaussian random numbers
     pz = px*px + py*py;
     Float_t random = gRandom->Rndm(1);
     hpx->Fill(px);
     hpxpy->Fill(px,py);
     hprof->Fill(px,pz);
     ntuple->Fill(px,py,pz,random,i);
  }

  // Save all objects in this file
  hfile.Write();

  // Close the file. Note that this is automatically done when you leave
  // the application.
  hfile.Close();

  return 0;
}
```

Root > .x hsimple.C

Root > .x hsimple.C+

TLatex formula emulator

```
{
   //example illustrating a TPaveText with Latex inside
   gROOT->Reset();
   TCanvas c1("c1");
   TPaveText pt(.05,.1,.95,.8);

   pt.AddText("#frac{2s}{#pi#alpha^{2}}  #frac{d#sigma}{dcos#theta} (e^{+}e^{-} #rightarrow f#bar{f} ) = \
 #left| #frac{1}{1 - #Delta#alpha} #right|^{2} (1+cos^{2}#theta)");

   pt.AddText("+ 4 Re #left{ #frac{2}{1 - #Delta#alpha} #chi(s) #[]{#hat{g}_{#nu}^{e}#hat{g}_{#nu}^{f} \
(1 + cos^{2}#theta) + 2 #hat{g}_{a}^{e}#hat{g}_{a}^{f} cos#theta) } #right}");

   pt.AddText("+ 16#left|#chi(s)#right|^{2}
 #left[(#hat{g}_{a}^{e}^{2} + #hat{g}_{v}^{e}^{2})
 (#hat{g}_{a}^{f}^{2} + #hat{g}_{v}^{f}^{2})(1+cos^{2}#theta)
 + 8 #hat{g}_{a}^{e} #hat{g}_{a}^{f} #hat{g}_{v}^{e} #hat{g}_{v}^{f}cos#theta#right] ");

   pt.SetLabel("Born equation");
   pt.Draw();
}
```

Born equation

$$\frac{2s}{\pi\alpha^2}\frac{d\sigma}{dcos\theta}(e^+e^- \rightarrow f\bar{f}) = \left|\frac{1}{1-\Delta\alpha}\right|^2 (1+\cos^2\theta)$$

$$+ 4\,\mathrm{Re}\left\{\frac{2}{1-\Delta\alpha}\chi(s)\left[\hat{g}_\nu^e\hat{g}_\nu^f(1+\cos^2\theta) + 2\,\hat{g}_a^e\hat{g}_a^f\cos\theta\right]\right\}$$

$$+ 16|\chi(s)|^2\left[(\hat{g}_a^{e^2} + \hat{g}_v^{e^2})(\hat{g}_a^{f^2} + \hat{g}_v^{f^2})(1+\cos^2\theta) + 8\,\hat{g}_a^e\,\hat{g}_a^f\,\hat{g}_v^e\,\hat{g}_v^f\cos\theta\right]$$

TPrincipal

by Christian Holm Christensen

- Principal Components Analysis class
- Translation in C++ of LINTRA (cernlib)
- Extensive documentation at URL:
  - http://root.cern.ch/root/html/TPrincipal.html
- Tutorial example
  - $ROOTSYS/tutorials/principal.C

TMultiDimFit

by Christian Holm Christensen

- MultiDimensional Parameterisation and Fit
- Translation in C++ of MUDIFI
- Extensive documentation at URL:
  - http://root.cern.ch/root/html/TMultiDimFit.html
- Tutorial example
  - $ROOTSYS/tutorials/multidimfit.C

New Histogram type TH1K

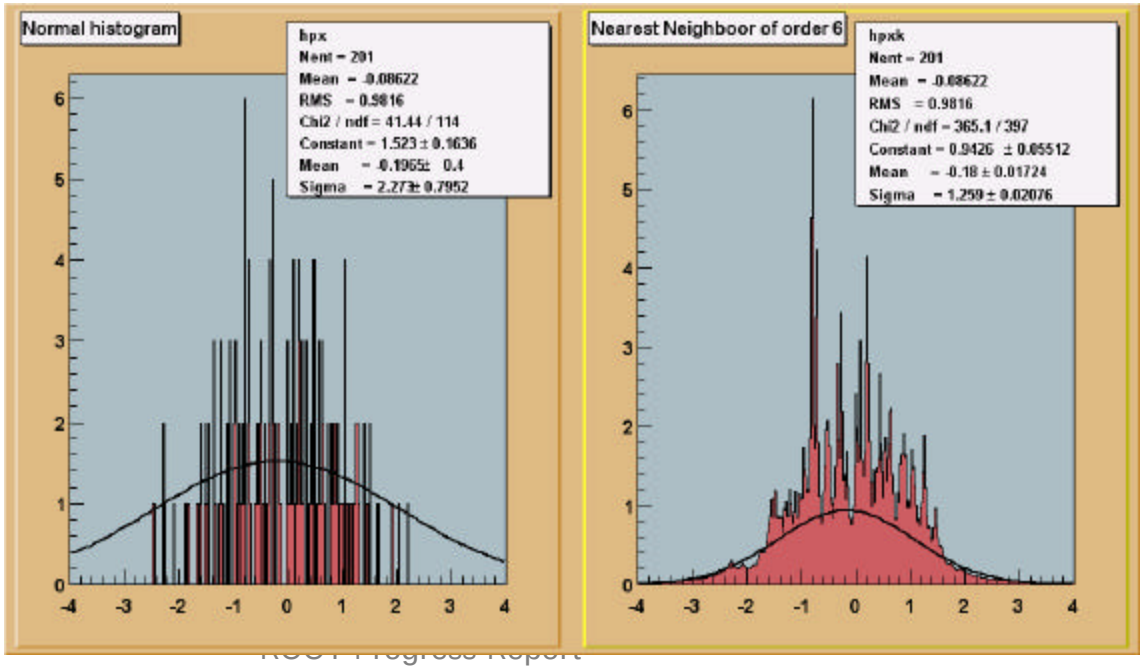
by Victor Perevoztchikov

TH1K class supports the nearest K Neighbors method, widely used in cluster analysis.

This method is especially useful for small statistics.
In this method : DensityOfProbability ~ 1/DistanceToNearestKthNeighbor

TH1K::TH1K(name,title,nbins,xlow,xup,K=0)
differs from TH1F only by "K"
K - is the order of K Neighbors method, usually >=3
K = 0, means default, where K is selected by TH1K in such a way
that DistanceToNearestKthNeighbor > BinWidth and K >=3

A TH1F and TH1K with 400 bins and 200 entries



Version 3.00

- Constness in many classes (TObject)
- TStreamerInfo
- rootcint supports more complex C++
- TFile::ShowStreamerInfo
- TFile::MakeProject
- Long_t portable format
- Parallel Sockets
- Many improvements in GUI
- new TTreeViewer
- new classes TBits TMultiGraph TXTRu, TH1K
- Font size in pixels (precision 3)
- new test suite bench
- many new tutorials

ROOT classes are const aware

- We had to do it. Helps in making your functions const
- Has generated some backward incompatibility
- Warning: compilers are poor in reporting potential conflicts when redefining functions in derived classes

```
virtual const char *ClassName() const;
virtual TObject    *Clone(const char *newname="") const;
virtual Int_t       Compare(const TObject *obj) const;
virtual TObject    *DrawClone(Option_t *option="") const;
virtual TObject    *FindObject(const char *name) const;
virtual TObject    *FindObject(const TObject *obj) const;
virtual Option_t   *GetDrawOption() const;
virtual UInt_t      GetUniqueID() const;
virtual const char *GetName() const;
virtual const char *GetIconName() const;
virtual char       *GetObjectInfo(Int_t px, Int_t py) const;
virtual ULong_t     Hash() const;
virtual Bool_t      InheritsFrom(const char *classname) const;
virtual Bool_t      InheritsFrom(const TClass *cl) const;
virtual Bool_t      IsFolder() const;
virtual Bool_t      IsEqual(const TObject *obj) const;
virtual Bool_t      IsSortable() const { return kFALSE; }
        Bool_t      IsOnHeap() const { return TestBit(kIsOnHeap); }
        Bool_t      IsZombie() const { return TestBit(kZombie); }
virtual void        ls(Option_t *option="") const;
virtual void        Print(Option_t *option="") const;
```

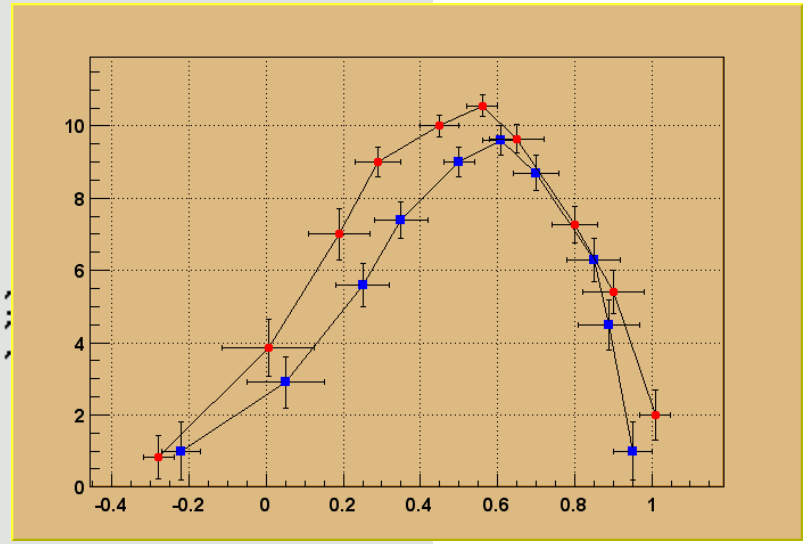
Example TObject

New class TBits

```
const Int_t N = 100000;
TBits bits(N);
for (Int_t i=0;i<10;i++) {
   Int_t bit = (Int_t)(gRandom->Rndm()*N);
   bits.SetBitNumber(bit,1);
}
bits.Print();
bits.Draw();
TFile f("bits.root","recreate");
bits.Write("bits");
```

New class TMultiGraph

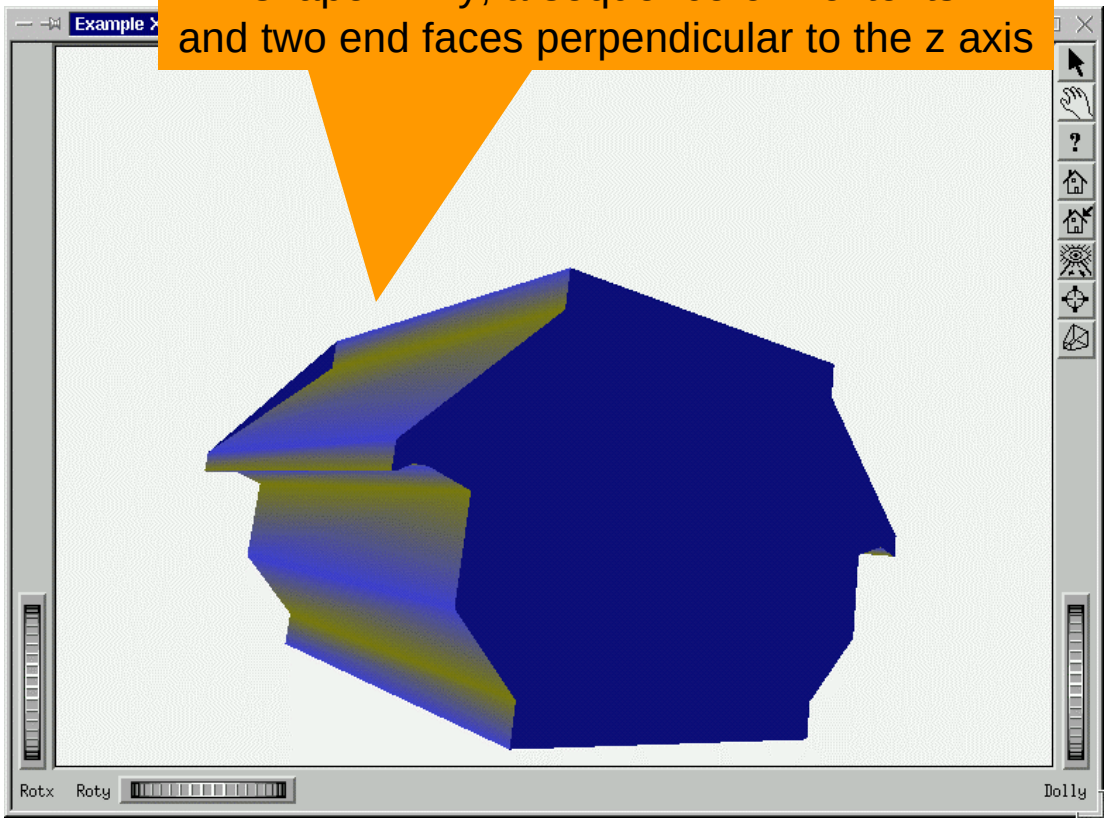
```
{
   c1 = new TCanvas("c1","gerrors2",200,10,700,500);
   c1->SetFillColor(42);
   c1->SetGrid();

      // draw a frame to define the range
   TMultiGraph *mg = new TMultiGraph();
   c1->GetFrame()->SetFillColor(21);
   c1->GetFrame()->SetBorderSize(12);

      // create first graph
   Int_t n1 = 10;
   Double_t x1[]  = {-0.22, 0.05, 0.25, 0.35, 0.5, 0.61,
   Double_t y1[]  = {1,2.9,5.6,7.4,9,9.6,8.7,6.3,4.5,1};
   Double_t ex1[] = {.05,.1,.07,.07,.04,.05,.06,.07,.08,
   Double_t ey1[] = {.8,.7,.6,.5,.4,.4,.5,.6,.7,.8};
   gr1 = new TGraphErrors(n1,x1,y1,ex1,ey1);
   gr1->SetMarkerColor(kBlue);
   gr1->SetMarkerStyle(21);
   mg->Add(gr1);

      // create second graph
   Int_t n2 = 10;
   Float_t x2[]  = {-0.28, 0.005, 0.19, 0.29, 0.45, 0.56,0.65,0.80,0.90,1.01};
   Float_t y2[]  = {0.82,3.86,7,9,10,10.55,9.64,7.26,5.42,2};
   Float_t ex2[] = {.04,.12,.08,.06,.05,.04,.07,.06,.08,.04};
   Float_t ey2[] = {.6,.8,.7,.4,.3,.3,.4,.5,.6,.7};
   gr2 = new TGraphErrors(n2,x2,y2,ex2,ey2);
   gr2->SetMarkerColor(kRed);
   gr2->SetMarkerStyle(20);
   mg->Add(gr2);

   mg->Draw("alp");
}
```



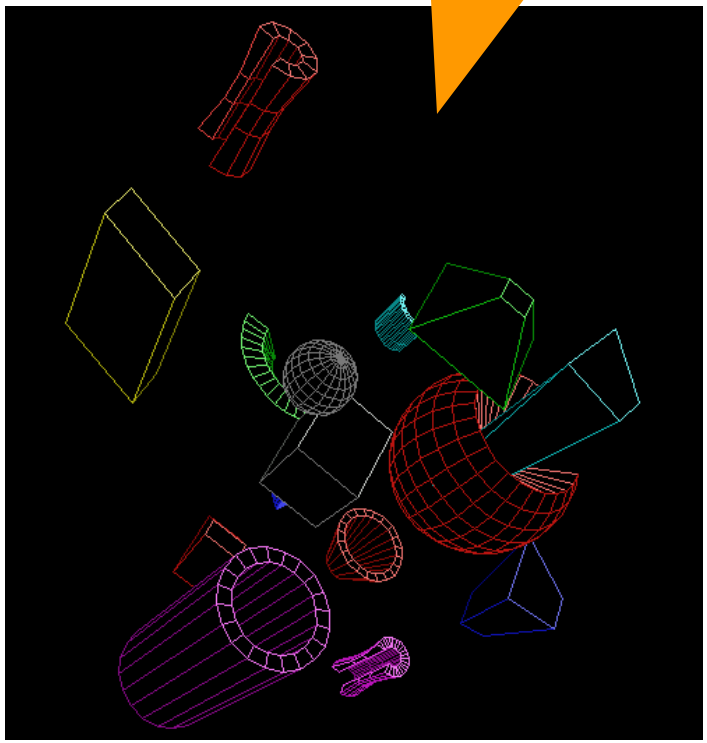
New geometry class TXTRU

by Robert Hatcher

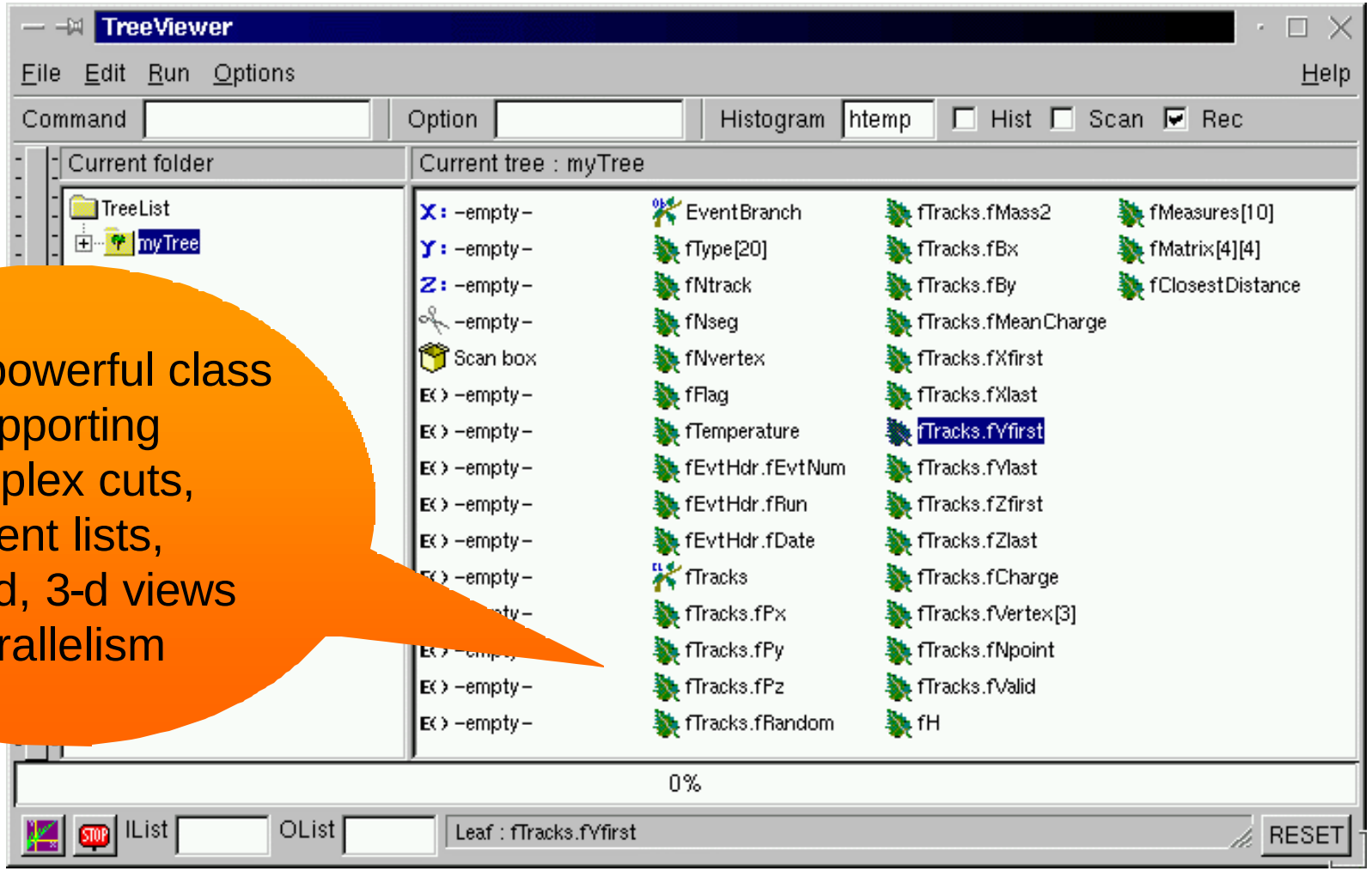
A TXTRU example in OpenInventor
XTRU is a poly-extrusion with fixed outline shape in x-y, a sequence of z-extents and two end faces perpendicular to the z axis



The 15 GEANTshapes seen in the X3D viewer



The Tree Viewer & Analyzer



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

Version 3.01

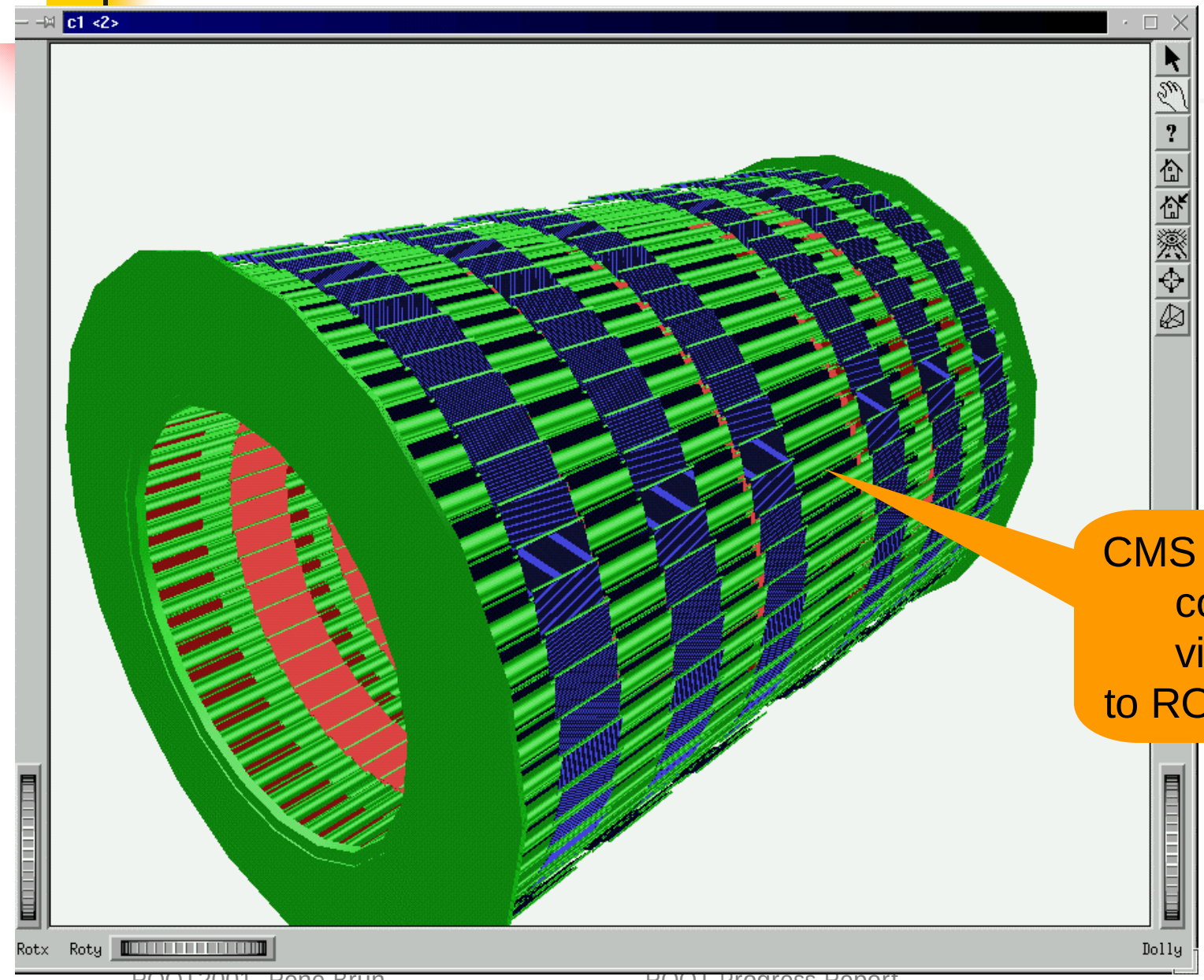
- New class TBranchElement
  - new branch style by default
- split/no-split mode symmetric
- Most restrictions in split mode removed
- TClonesArray accept more complex classes
- Tree Friends

- Many improvements in TTreeFormula
  - (thanks Philippe Canal)
- Trees can be generated from Folders
- Can read/query Trees without the classes
- Open Inventor Interface

See talk on Trees

See talk on Folders

ROOT + OpenInventor



CMS with Geant3
converted
via g2root
to ROOT TNodes

ROOT and the WEB

- An Apache Web-server plug-in module is being developed by Valeriy OnuChin.
- Provides interactive access to ROOT files, CINT macros and all the graphics. Web pages generated on the fly.
- Interesting alternative to PHP using C++ as an embedded scripting system with full access to user classes dynamically
- See talk by Valeriy Friday morning

Evolution of ROOT I/O

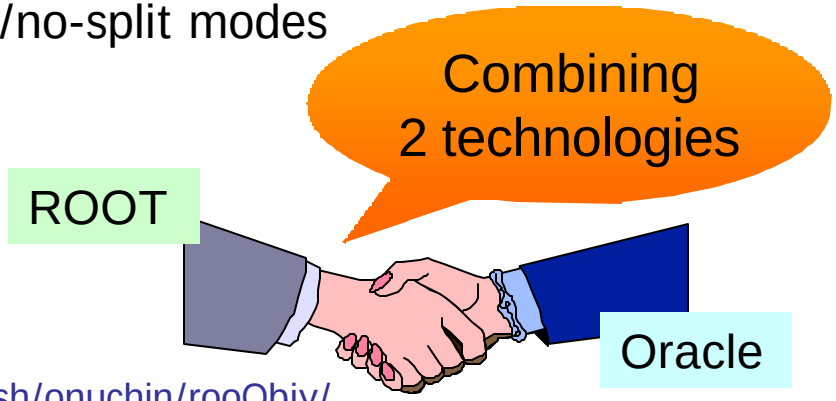
- Hand-written Streamers
- Streamers generated via rootcint
- Support for Class Versions
- Support for ByteCount
- Several attempts to introduce automatic class evolution
- Persistent class Dictionary written to files
- rootcint modified to generate automatic Streamers
- can generate code for “DataObjects” 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

3.00

3.01

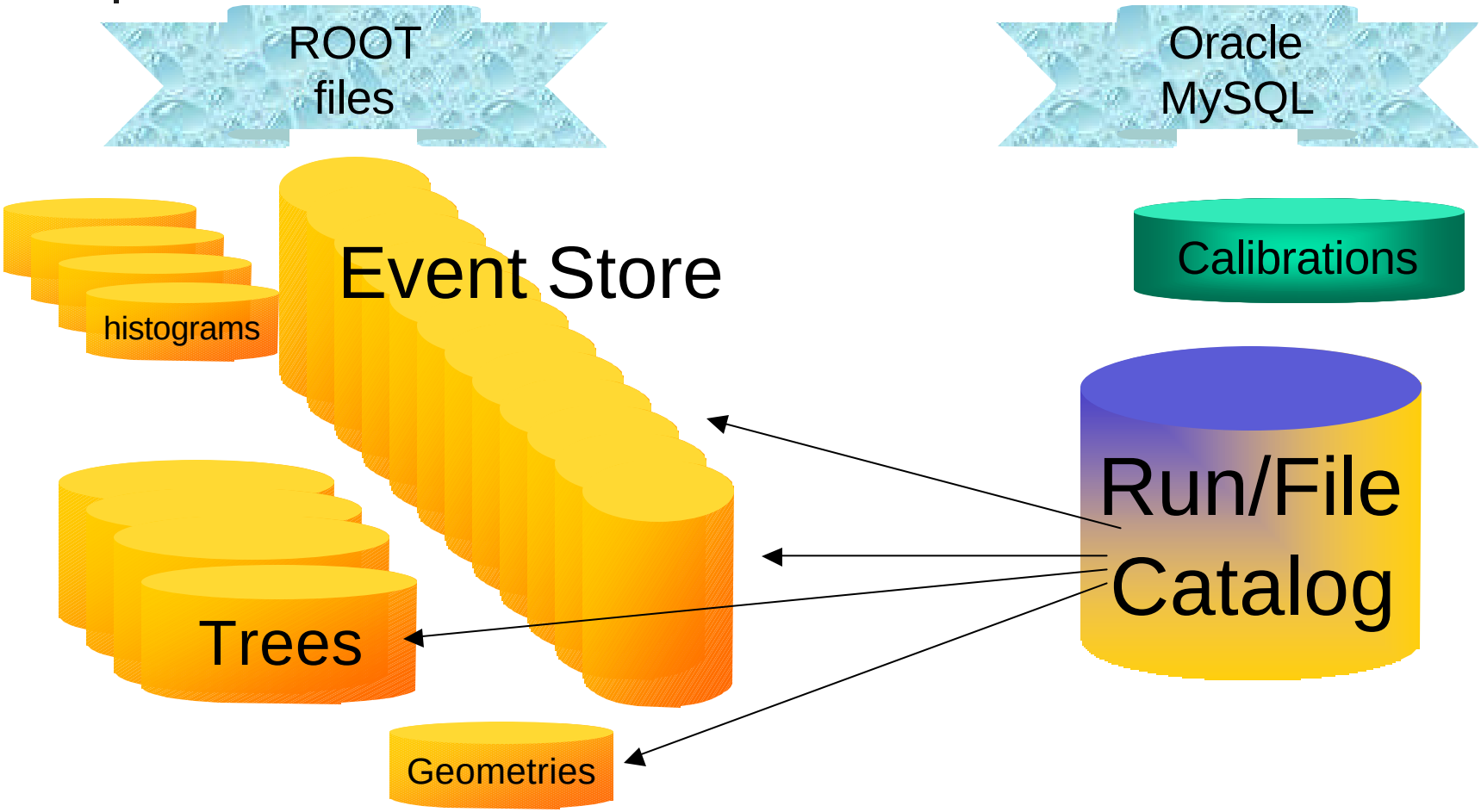
ROOT Data Base Model

- Put Event write-once data in an object store
  - via object.Write() or Trees in split/no-split modes
- Use a RDBMS for :
  - Run/Event catalogs
  - Geometry, calibrations
  - eg with ROOT<->Oracle interface
    - http://www.phenix.bnl.gov/WWW/publish/onuchin/rooObjy/



- Use ROOT split/no-split mode for phys analysis
- Use PROOF for large-scale distributed physics analysis
- Integration with the GRID model

ROOT + RDBMS Model



Old Streamers in 0.90/08

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

private:
  Int_t         fNbins;
  Axis_t        fXmin;
  Axis_t        fXmax;
  TArrayF       fXbins;
  Char_t       *fXlabels;
```

rootcint

```
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);
   }
}
```

Old Streamers in 2.25

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

private:
   Int_t        fNbins;
   Axis_t       fXmin;
   Axis_t       fXmax;
   TArrayF      fXbins;
   Char_t      *fXlabels;
   Int_t        fFirst;
   Int_t        fLast;
   TString      fTimeFormat;
   Bool_t       fTimeDisplay;
   TObject     *fParent;
```

rootcint

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

New Streamers in 3.00

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

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

rootcint

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

   if (R__b.IsReading()) {
      UInt_t R__s, R__c;
      Version_t R__v = R__b.ReadVersion(&R__s, &R__c);
      if (R__v > 5) {
         TAxis::Class()->ReadBuffer(R__b, this, R__v, R__s, R__c);
         return;
      }
      //====process old versions before automatic schema evolution
      ....
      //====end of old versions

   } else {
      TAxis::Class()->WriteBuffer(R__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

Self-describing files

- Dictionary for persistent classes written to the file when closing the file.
- ROOT files can be read by foreign readers (see presentation on JavaRoot (Tony slides)
- 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++");
```

StreamerInfo

```
root [8] f.Map()
20010523/120223  At:64        N=124       TFile
20010523/120223  At:188       N=111       TBasket         CX = 30.44
20010523/120223  At:299       N=227       TBasket         CX = 14.87
...
...
20010523/120230  At:1739750   N=212       TBasket         CX = 12.88
20010523/120230  At:1739962   N=116       TBasket         CX = 45.56
20010523/120230  At:1740078   N=62370     TTree           CX = 25.47
20010523/120230  At:1802448   N=9101      ATLFast         CX =  7.39
20010523/120230  At:1811549   N=14061     StreamerInfo    CX =  4.64
20010523/120230  At:1825610   N=182       KeysList
20010523/120231  At:1825792   N=73        FreeSegments
20010523/120231  At:1825865   N=1         END
```

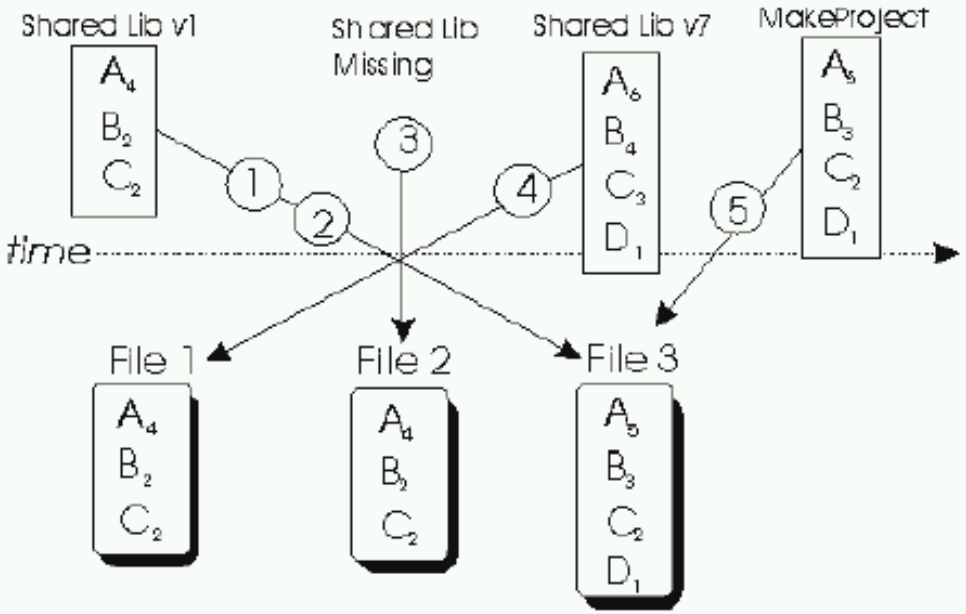
The description of
all classes
in a file
is written
in one single record
when the file is closed
StreamerInfo

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

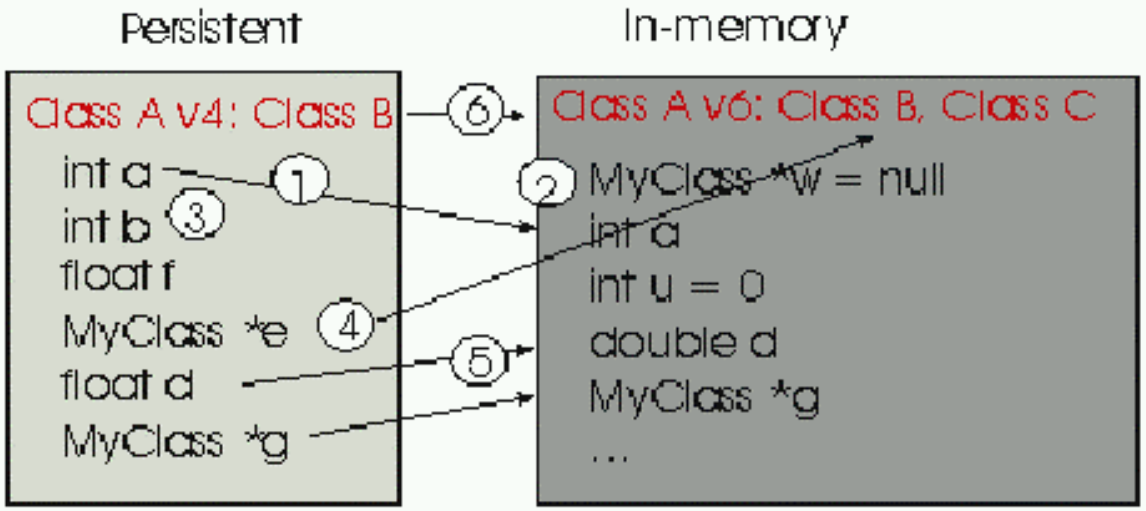


1) An old version of a shared library and a file with new class definitions. This can be the case when someone has not updated the library and is reading a new file.
2) Reading a file with a shared library that is missing a class definition (i.e. missing class D).
3) Reading a file without any class 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



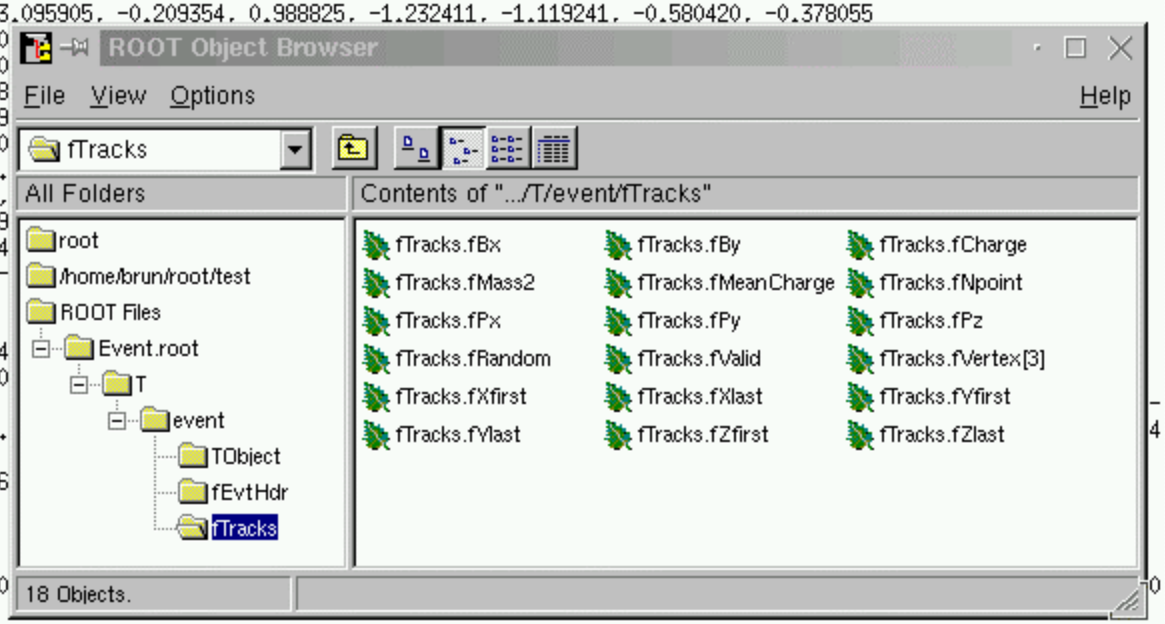
Missing Classes

The system warns you when opening a file and the class library is missing

```
root [0] TFile f("atlfast.root")
Warning in <TClass::TClass>: no dictionary for class TMCParticle is available
Warning in <TClass::TClass>: no dictionary for class ATLFMuon is available
Warning in <TClass::TClass>: no dictionary for class ATLFElectron is available
Warning in <TClass::TClass>: no dictionary for class ATLFPhoton is available
Warning in <TClass::TClass>: no dictionary for class ATLFJet is available
Warning in <TClass::TClass>: no dictionary for class ATLFMisc is available
Warning in <TClass::TClass>: no dictionary for class ATLFTrigger is available
Warning in <TClass::TClass>: no dictionary for class ATLFTrack is available
Warning in <TClass::TClass>: no dictionary for class ATLFast is available
Warning in <TClass::TClass>: no dictionary for class ATLFMCMaker is available
Warning in <TClass::TClass>: no dictionary for class ATLFMaker is available
Warning in <TClass::TClass>: no dictionary for class ATLFClusterMaker is available
Warning in <TClass::TClass>: no dictionary for class ATLFCluster is available
Warning in <TClass::TClass>: no dictionary for class ATLFMuonMaker is available
Warning in <TClass::TClass>: no dictionary for class ATLFElectronMaker is available
Warning in <TClass::TClass>: no dictionary for class ATLFPhotonMaker is available
Warning in <TClass::TClass>: no dictionary for class ATLFJetMaker is available
Warning in <TClass::TClass>: no dictionary for class ATLFMiscMaker is available
Warning in <TClass::TClass>: no dictionary for class ATLFTriggerMaker is available
Warning in <TClass::TClass>: no dictionary for class ATLFTrackMaker is available
Warning in <TClass::TClass>: no dictionary for class ATLFHistBrowser is available
root [1]
```

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]
```



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.Draw("fPx")

root [2] TLeaf *leaf = T.GetLeaf("fNtrack")

root [3] T.GetEntry(123)

root [4] leaf->GetValue()
(const Double_t)5.99000000000000000e+02

root [5] leaf->GetBranch()->GetEntry(89)

root [6] leaf->GetValue()
(const Double_t)6.07000000000000000e+02
```

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
   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

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

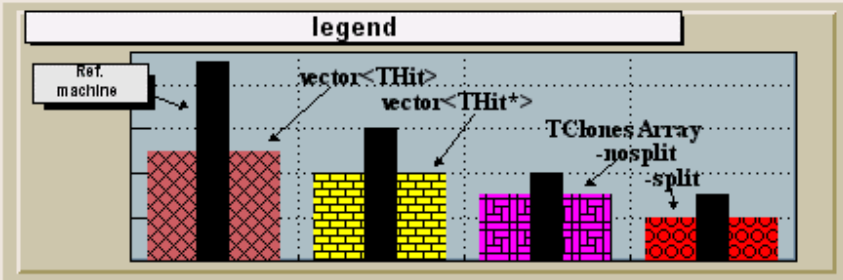
Time to read in seconds	comp 0	Reference	comp 1	Reference
vector<THit>	3.03	2.29	4.24	3.67
vector<THit*>	3.01	2.10	4.28	3.27
TClonesArray(TObjHit)	1.34	1.53	1.88	2.14
TClonesArray(TObjHit) split	1.35	1.35	1.88	1.94
Total CPU time	82.14	76.33		
Estimated ROOTMARKS	185.85	200.00		

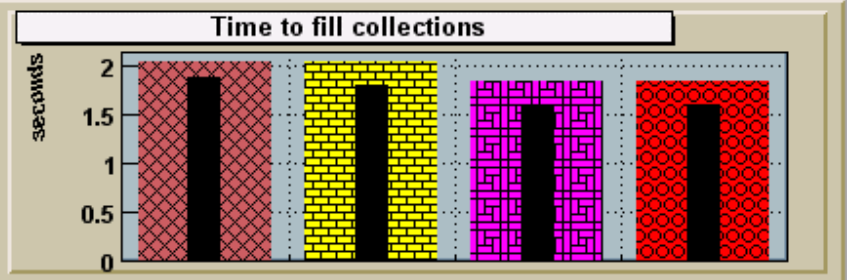
Better compression with TClonesArray

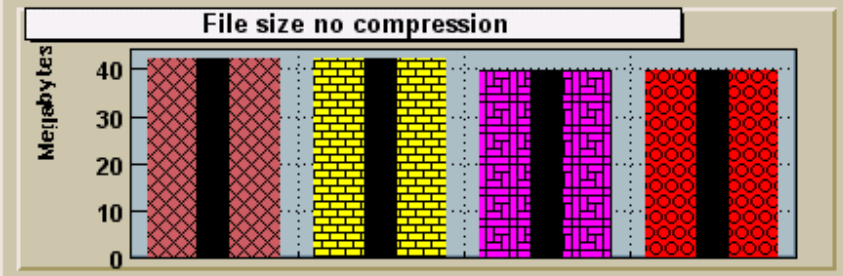
Better write with TClonesArray

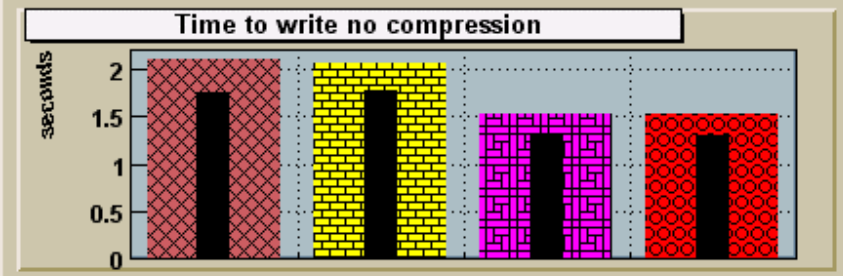
Much better read with TClonesArray

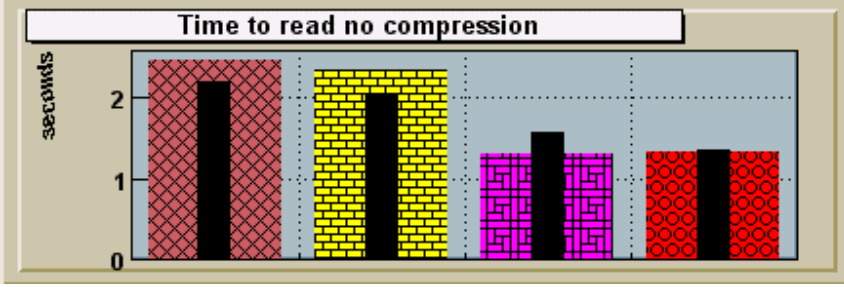
Comparing STL vector with TClonesArray: Root 3.01/03
Linux pcnotebrun 2.2.19 #1 Tue May 15 12:33:56 MEST 2001 i6
Reference machine pcnotebrun.cern.ch RedHat Linux 6.1
(Pentium III 650 Mhz 256 Mbytes RAM, IDE disk)
(send your results to rootdev@root.cern.ch)

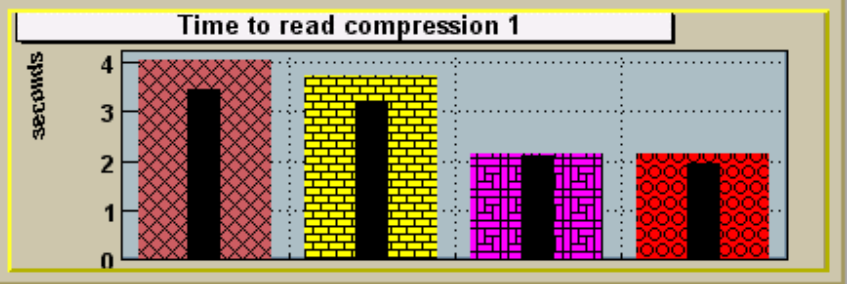












Makefiles

- 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
```

Download source, Binaries

- Intel x86 Linux for Redhat 7.0 (glibc 2.2pre) and gcc 2.96 (patched), version 3.00/06 (7.9 MB). NEW
- Intel x86 Linux for Mandrake7.2 and gcc2.95, version 3.01/00 (7.9 MB). NEW
- Intel x86 Linux for Redhat 6.1 (glibc 2.1) and egcs1.1.2, version 3.01/03 (8.6 MB). NEW
- Intel x86 Linux for Redhat 5.0/5.1/5.2 (glibc) and egcs 1.1.1, version 3.01/03 (8.3 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 3.01/03 (12.3 MB). NEW
- Compaq Alpha OSF1 with cxx 6.2, version 3.01/03 (8.9 MB). NEW
- Compaq Alpha OSF1 with egcs 1.1.2, version 3.01/03 (10.1 MB). NEW
- Compaq Alpha Linux with egcs 1.1.2, version 3.00/06 (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 3.01/03 (11.2 MB, works only on AIX 4.3). NEW
- Sun SPARC Solaris 2.6 with CC5.0, version 3.01/03 (9.7 MB). NEW
  It cannot be used with Solaris 2.7 or 2.8 even using the same compiler version. You must recompile from the source on these two systems.
- Sun SPARC Solaris 2.8 with CC5.2, version 3.01/03 (9.7 MB). NEW
  It cannot be used with Solaris 2.6 even using the same compiler version. You must recompile from the source on these two systems.
- SGI IRIX 6.5 with CC, version 3.01/03 (compiled with –n32) (10.2 MB). NEW
- SGI IRIX 6.5 with g++ 2.95.2, version 3.01/03 (11.4 MB). NEW
- SGI IRIX 6.5 with KCC, version 3.01/03 (9.8 MB). NEW
- LinuxPPC/2000 (glibc 2.1) gcc 2.95, version 3.01/02 (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 3.01/03 (good old tar file) (9.3 MB). NEW
- Windows/NT/95/98 with VC++ 6.0,compiled with debug info, version 3.01/03 (good old tar file) (9.3 MB). NEW
- Windows/NT/95/98 with VC++ 6.0, version 3.01/03 (built with InstallShield) (9.1 MB). NEW

20 binary tar balls + source

ROOT Downloads

114,000 binaries download

650,000 clicks per month

19,000 docs in 6 months

2200 reg users in roottalk