Recent Developments in ROOT I/O
Support for foreign classes

LCG meeting

CERN- 6 June

René Brun

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

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 (StreamerInfo) 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
- Support for foreign classes

3.00

3.01

3.02

New

3.03

2002

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;
```

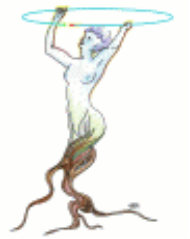
Point3DBase.h

```
#include "PV3DBase.h"
#include "Point2DBase.h"
#include "Vector3DBase.h"
#include ”Rtypes.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)
```

Basic3DVector.h

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

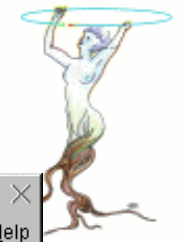
Writing CMS PSimHit objects

```
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");

}
```

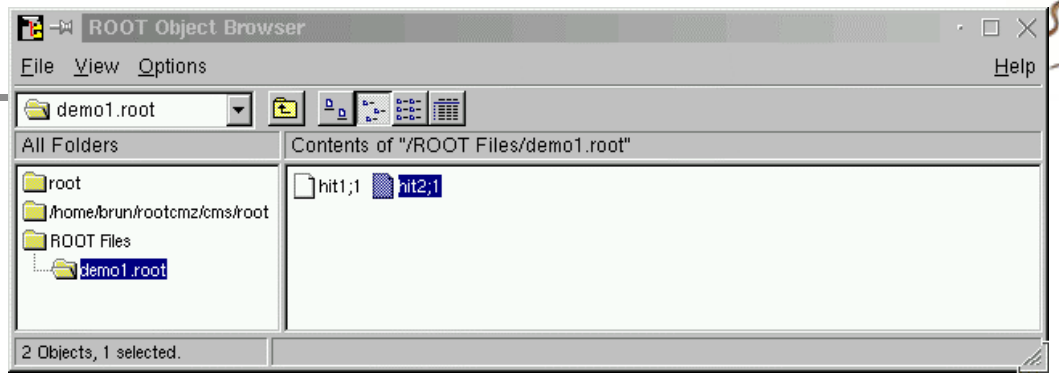
Reading CMS PSimHit objects

```
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

ClassDef/ClassImp changes

We have redesigned the ClassDef/ClassImp facility to drastically reduce the number of macros necessary. We are down to 3 macros for all the situations. The only macro required in now ClassDef. In particular, ClassDefT and all its variation are now obsolete. ClassImp (and templateClassImp for class template) are not compulsory anymore and are now used only to record the location of the implementation file. One future extension will be to allow the registering of more than one implementation file.

This upgrade is intended to be backward compatible ... __unless__ you have implemented your own ClassImp type of macro.

Note that it requires a complete regeneration of ALL the dictionaries.

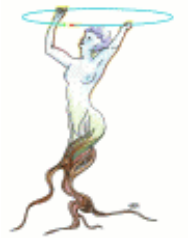
Point3DBase.h

```
#include "PV3DBase.h"
#include "Point2DBase.h"
#include "Vector3DBase.h"
#include ”Rtypes.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)
{}

…….
ClassDef(Point3DBase,1)
};
```

Basic3DVector.h

```
#include "Basic2DVector.h"
#include <iosfwd>
#include <cmath>
#include "Rtypes.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;

ClassDef(Basic3DVector,1)
};
```

Support for foreign classes

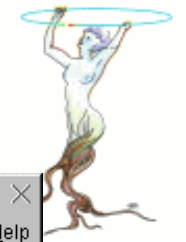


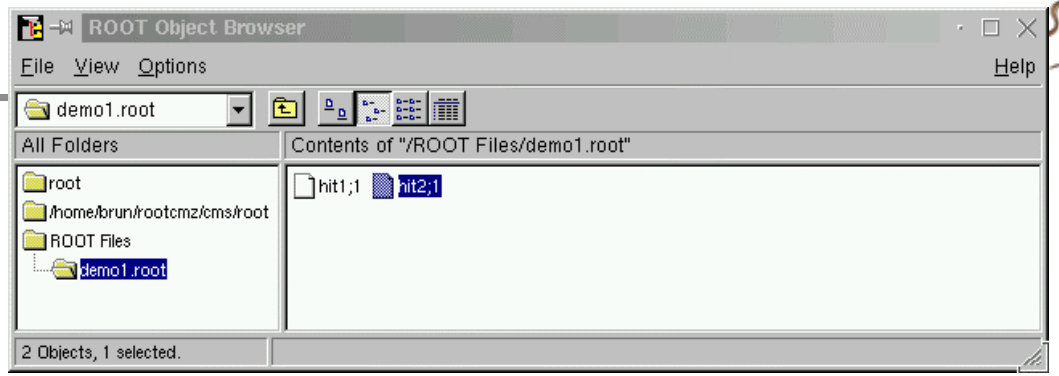
- Foreign classes (not instrumented with ROOT includes, ClassDef, etc) can be parsed by a new version of rootcint to produce the necessary code for computing the members offsets.
- The TStreamerInfo class has also some mods to digest foreign classes.
- Current implementation using a shadow class
  - Vincenzo (define public private) was also implemented but could not be used with VC++
  - Victor currently testing a new idea that could simplify the whole process if it works everywhere

Point3DBase.h

```
#include "PV3DBase.h"
#include "Point2DBase.h"
#include "Vector3DBase.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)
{}

…….
};
```

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

```
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
```

```
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
```

STL support for foreign classes

- Several recent improvements by Victor in rootcint.
- In particular Victor has added support for all STL containers in foreign classes.
- Currently the code to support STL is generated in rootcint. We hope to provide native support in TStreamerInfo for vector<myClass> like we do for TClonesArray.