Subject: [FIXED] Memory leak with TClonesArray and std::vector
Posted by Klaus Götzen on Fri, 23 Aug 2013 07:36:11 GMT
View Forum Message <> Reply to Message

Hi,

this is more a ROOT related question, but maybe somebody from this club has an idea or knows about it.

The problem is that I observed quite a severe memory leak (just looking with 'top' how memory consumption of the job develops), apparently because the RhoCandidate stores the daughter pointers in a std::vector<RhoCandidate*>.

I isolated the problem in a short ROOT macro looking like:

```
#include "TClonesArray.h"
#include <vector>
#include <iostream>

// very simple class to be used with a TClonesArray
class MyClass: public TObject
{
public:
   MyClass() {};
   ~MyClass() {};
   std::vector<int> fVec;
   ClassDef(MyClass,1)
};

void testtca(int num=100000)
{
   TClonesArray *tca = new TClonesArray("MyClass",1100);

   for (int i=0;i<num;++i)
   {
      if ((i%100)==0) cout <<i<<endl;
 tca->Clear();

 for (int  j=0;j<1000;++j)
       {
    MyClass *c= new ((*tca)[j]) MyClass();
    c->fVec.push_back(42); // <-  THIS IS THE BAD GUY!!1
    c->fVec.clear();
 }
   }
}
```

When you run this with 'root -l -b -q testtca.C+' you will see the memory growing very!! fast (be careful, you might want to reduce num).

When you comment out the line with 'push_back(42)' it works perfectly well like you would expect it from a TClonesArray. The interesting part is, that even the 'fVec.clear()' directly after the push_back does not help at all to avoid the problem (most likely because clear() does not really release the allocated memory of the vector). And I think even an appropriate cleanup in the destructor wouldn't really help, since the objects in a TCA are not deleted anyway, right?

I didn't find any way to fix this leak up to now. Does somebody here has an idea how to avoid that problem? Or a solution for a dynamic container for pointers different from std::vector?

Best,
Klaus
