hddm2root

Oct. 31, 2012

David Lawrence JLab

HDDM

- The **H**all-**D** **D**ata **M**odel (**HDDM***) package provides a way to store and retrieve data structures on disk
- The data model is specified using an XML file
- C and C++ interfaces are generated from the XML using the HDDM tools
- Different data models may be used and distinguished via the "class"
- Simulated data for GlueX is written to hddm files of class "s"
- Reconstructed summary data files are now being written to hddm files of class "r"

**HDDM is written an maintained by Richard Jones of UConn*

hddm2root

- hddm2root is a program that will create a tool that can convert data from an hddm file into a ROOT file
- The XML that specifies an HDDM class is read in and C++ code is generated for a tool specific to that HDDM class
- The XML can be taken either from the original XML specification file, or from an existing hddm file

Example

XML specification for HDDM class "x"

```
<?xml version="1.0" encoding="iso-8859-1" standalone="no" ?>
<HDDM xmlns="http://www.gluex.org/hddm" class="x" version="1.0.0">

    <event eventNo="int" runNo="int">
        <vertex maxOccurs="unbounded">
            <particle maxOccurs="unbounded" type="int" px="float" py="float" pz="float"/>
        </vertex>
    </event>

</HDDM>
```

Running hddm2root

```
>hddm2root example.xml
writing hddm_root_generated/event_t.h
writing hddm_root_generated/particle_t.h
writing hddm_root_generated/vertex_t.h
Writing hddm_root_generated/hddm_root_CopyRoutines.h
Writing hddm_root_generated/hddm_root_CopyRoutines.cc
Writing hddm_root_generated/hddm2root_x.cc

Code generation complete. Issue the following
to build the hddm2root_x tool:

  make -C hddm_root_generated

The tool will be left as hddm_root_generated/hddm2root_x
>
```

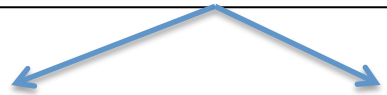
Running "make" as indicated will:

1. Cause *rootcint* to be run for each data object header file in order to generate a ROOT dictionary file

2. Compile the generated dictionary files as well as the *hddm2root* generated files into the *hddm2root_x* executable

XML specification for HDDM class "x"

```xml
<?xml version="1.0" encoding="iso-8859-1" standalone="no" ?>
<HDDM xmlns="http://www.gluex.org/hddm" class="x" version="1.0.0">

    <event eventNo="int" runNo="int">
        <vertex maxOccurs="unbounded">
            <particle maxOccurs="unbounded" type="int" px="float" py="float" pz="float"/>
        </vertex>
    </event>

</HDDM>
```



vertex_t.h

```
#include<vector>
using namespace std;

#include<Rtypes.h>
#include<TObject.h>

#ifndef _vertex_t_
#define _vertex_t_
typedef int Particle_t;

#include "particle_t.h"

class vertex_t:public TObject{
    public:

        vector<particle_t> particles;

        ClassDef(vertex_t,1)
};

#endif
```

particle_t.h

```
#include<vector>
using namespace std;

#include<Rtypes.h>
#include<TObject.h>

#ifndef _particle_t_
#define _particle_t_
typedef int Particle_t;


class particle_t:public TObject{
    public:

        float px;
        float py;
        float pz;
        int type;

        ClassDef(particle_t,1)
};

#endif
```

Examples of generated copy routines

```
void CopyParticle(particle_t &particle, class Particle &particle_hddm, bool hddm_invalid)
{
    if(hddm_invalid)return; // FIXME!!!

    particle.px = particle_hddm.getPx();
    particle.py = particle_hddm.getPy();
    particle.pz = particle_hddm.getPz();
    particle.type = particle_hddm.getType();
}

void CopyVertex(vertex_t &vertex, class Vertex &vertex_hddm, bool hddm_invalid)
{
    if(hddm_invalid)return; // FIXME!!!


    ParticleList &particles = vertex_hddm.getParticles();
    ParticleList::iterator iter_particle;
    for(iter_particle=particles.begin(); iter_particle!=particles.end(); iter_particle++){
        particle_t a;
        CopyParticle(a, *iter_particle);
        vertex.particles.push_back(a);
    }
}
```

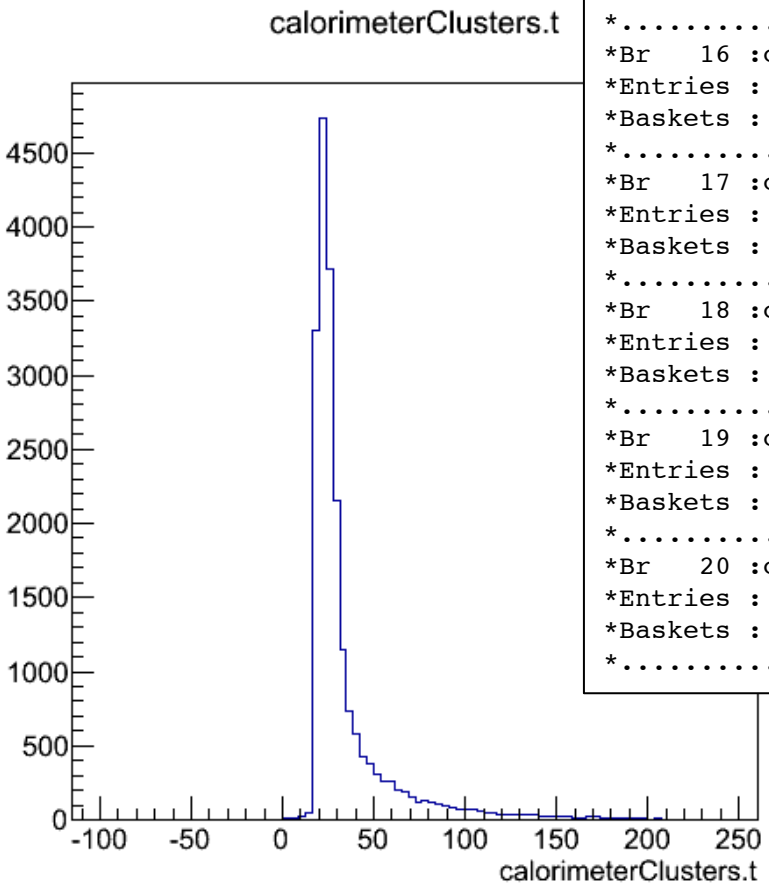
Example of generated clear routine

```
void ClearEvent(event_t &event)
{

    event.eventNo = 0;
    event.runNo = 0;
    event.vertexs.clear();
}
```

Tool calls the Clear and Copy routines for top-level objects

```
    // Loop over events
    unsigned int N = 0;
    while(!ifs.eof()){
        try{
            HDDM xrec;
            istr >> xrec;
            ClearEvent(*event);
            CopyEvent(*event, xrec.getEvent());

            t->Fill();
            if(++N%10 == 0){cout<<" "<<N<<" events processed    \r"; cout.flush();}
            if(MAX_EVENTS>0 && N>=MAX_EVENTS)break;
        }catch(...){
            break;
        }
    }
```

root generated from REST data




```
*.....................................................................*
*Br   15 :calorimeterClusters.t : Float_t t[calorimeterClusters_]      *
*Entries :     2896 : Total  Size=      93426 bytes  File Size  =      78485 *
*Baskets :        4 : Basket Size=      32000 bytes  Compression=   1.18     *
*.....................................................................*
*Br   16 :calorimeterClusters.terr : Float_t terr[calorimeterClusters_]    *
*Entries :     2896 : Total  Size=      93450 bytes  File Size  =      45032 *
*Baskets :        4 : Basket Size=      32000 bytes  Compression=   2.06     *
*.....................................................................*
*Br   17 :calorimeterClusters.tzcorr : Float_t tzcorr[calorimeterClusters_]  *
*Entries :     2896 : Total  Size=      93466 bytes  File Size  =       6501 *
*Baskets :        4 : Basket Size=      32000 bytes  Compression=  14.27     *
*.....................................................................*
*Br   18 :calorimeterClusters.x : Float_t x[calorimeterClusters_]      *
*Entries :     2896 : Total  Size=      93426 bytes  File Size  =      73393 *
*Baskets :        4 : Basket Size=      32000 bytes  Compression=   1.26     *
*.....................................................................*
*Br   19 :calorimeterClusters.xerr : Float_t xerr[calorimeterClusters_]    *
*Entries :     2896 : Total  Size=      93450 bytes  File Size  =      32041 *
*Baskets :        4 : Basket Size=      32000 bytes  Compression=   2.90     *
*.....................................................................*
*Br   20 :calorimeterClusters.xycorr : Float_t xycorr[calorimeterClusters_]  *
*Entries :     2896 : Total  Size=      93466 bytes  File Size  =       6501 *
*Baskets :        4 : Basket Size=      32000 bytes  Compression=  14.27     *
*.....................................................................*
```

```
> ls -sk dana_rest.hddm hddm2root_r.root
4114 dana_rest.hddm
4626 hddm2root_r.root
```

ROOT TSelector

Generated from T->MakeSelector(…)

```
103     UInt_t          comments_fUniqueID[kMaxcomments];   //[comments_]
104     UInt_t          comments_fBits[kMaxcomments];   //[comments_]
105     string          comments_text[kMaxcomments];
106     Int_t           eventNo;
107     Int_t           reactions_;
108     UInt_t          reactions_fUniqueID[kMaxreactions];   //[reactions_]
109     UInt_t          reactions_fBits[kMaxreactions];   //[reactions_]
110     Float_t         reactions_Ebeam[kMaxreactions];   //[reactions_]
111     string          reactions_jtag[kMaxreactions];
112     Int_t           reactions_targetType[kMaxreactions];   //[reactions_]
113     Int_t           reactions_type[kMaxreactions];   //[reactions_]
114   //vector<vertex_t> reactions_vertexs[kMaxreactions];
115     Float_t         reactions_weight[kMaxreactions];   //[reactions_]
116     Int_t           runNo;
117     Int_t           startHits_;
118     UInt_t          startHits_fUniqueID[kMaxstartHits];   //[startHits_]
119     UInt_t          startHits_fBits[kMaxstartHits];   //[startHits_]
120     Float_t         startHits_dE[kMaxstartHits];   //[startHits_]
121     string          startHits_jtag[kMaxstartHits];
122     Int_t           startHits_sector[kMaxstartHits];   //[startHits_]
123     Float_t         startHits_t[kMaxstartHits];   //[startHits_]
124     Int_t
125     UInt_t
126     UInt_t
127     Float_t
128     string
129     Float_t
```

```
4    <reconstructedPhysicsEvent eventNo="int" runNo="int">
5      <comment minOccurs="0" maxOccurs="unbounded" text="string"/>
6      <reaction maxOccurs="unbounded" minOccurs="0" jtag="string"
7                                      type="int" weight="float"
8                                      targetType="Particle_t"
9                                      Ebeam="float" Eunit="GeV">
10       <vertex maxOccurs="unbounded">
11         <origin t="float" vx="float" vy="float" vz="float" lunit="cm"/>
12         <product id="int" maxOccurs="unbounded" parentId="int" pdgtype="int">
13           <momentum E="float" px="float" py="float" pz="float" Eunit="GeV"
14                     punit="GeV/c"/>
15         </product>
16       </vertex>
17     </reaction>
```

ROOT will not handle too many nested containers!
Only one instance of this in REST, but many in class “s”.

This can be a real limitation.

Summary

- hddm2root tool can generate tools to convert hddm files of a specific class into ROOT trees
- Limitation of ROOT does not allow more than two levels of containers
- At the moment, inclusion of the HDDM I/O routines in libHDDM.a is required
  (Could be automated further with hddm-cpp)
- Source can be found here:
  https://halldsvn.jlab.org/repos/trunk/sim-recon/src/programs/Utilities/hddm2root