CODE SHARING AND THE EVIO PACKAGE

Elliott Wolin

CLAS12 Software Workshop

U of Richmond

25-May-2010

Code Sharing

- Many opportunities for sharing
  - Maybe with Halls A and C, but they are different
- CLAS12 and Hall D very similar
  - Online – almost identical
  - Offline – similar, but different computing models

Currently Shared Packages

- CLARA – SOA architecture
- JANA – multi-threaded analysis framework
- EVIO – binary I/O of in-memory tree model
- Event Display – bCNU plus customizations
- cMsg – generic publish/subscribe IPC
- CODA – JLab DAQ package
- EPICS – for online control systems
- RootSpy – display of distributed ROOT hist
- ROOT, MySQL, PHP, etc.

Current Joint Code Development Efforts

- Event display
  - CNU student working on Hall D event display
  - Developments could benefit CLAS12 as well
- RootSpy
  - CNU student from Yelena will start soon
  - Developments will benefit both halls

Online - Future Possibilities

- elog
- alarm system – BEAST?
- backup and restore – from ORNL?
- Archiver – from ORNL?
- EPICS displays – Labview? CSS?
- JavaIOC – in development
- online farm management
- online event processing/monitoring
- online databases
- controls database – IRMIS?

Offline - Future Possibilities

- Calibration constants database
- DST format
- Magnetic field storage on disk
- Track swimming in inhomogeneous field
- Kalman filter package
- Matrix package using SIMD instructions
- PWA analysis framework
- Geant4
- Geometry database and XML representation

EVIO

General Remarks

- Original C package just did binary buffer I/O
  - You had to manually set the bits and bytes
- Now implements in-memory object model
  - Includes auto-serialization to binary buffer
    - no more setting bits and bytes by hand!
  - C++ and Java (different in-memory models)
  - Implements XML-like tree in memory
    - C++ - Custom STL-based
    - Java – based on JTree
- Machine/architecture independent
- Automatically handles endian conversions

Basic Features

- Tree consists of container nodes and leaf nodes
- Container nodes only hold other nodes
- Leaf node contains array of one primitive type
  - int32_t, int 16_t, float, double, string, etc.
- Nodes can have 2- or 1-word header on disk
- Nodes with 2-word headers (“banks”) have user-settable int16_t “tag” and int8_t “num”

```
<event content="bank" data_type="0x10" tag="1" num="204">
  <bank1 content="segment" data_type="0x10" tag="2" num="1">
    <uint32 data_type="0x1" tag="2748">
      0xffffffff 0xfffffffe 0xfffffffd 0xfffffffc 0xfffffffb
      0xfffffffa 0xfffffff9 0xfffffff8 0xfffffff7 0xfffffff6
      0xfffffff5 0xfffffff4 0xfffffff3 0xfffffff2 0xfffffff1
      0xfffffff0 0xffffffef 0xffffffee 0xffffffed 0xffffffec
      0xffffffeb 0xffffffea 0xffffffe9 0xffffffe8 0xffffffe7
      0xffffffe6 0xffffffe5 0xffffffe4 0xffffffe3 0xffffffe2
    </uint32>
    <int32 data_type="0xb" tag="1">
      -1 -2 -3 -4 -5
      -6 -7 -8
    </int32>
    <float32 data_type="0x2" tag="2">
      -1.000000 -2.000000 -3.000000 -4.000000 -5.000000
      -6.000000 -7.000000 -8.000000 -9.000000 -10.000000
      -11.000000 -12.000000
    </float32>
```

```
#include <evioUtil.hxx>

int main(int argc, char **argv) {

  try {

    // create evio file channel object for reading, argv[1] is filename
    evioFileChannel chan(argv[1], "r");

    // open the file
    chan.open();

    // loop over events
    while(chan.read()) {

      // create tree from contents of file channel object
      evioDOMTree tree(chan);

      // print tree
      cout << tree.toString() << endl;

    }

    // eof reached...close file
    chan.close();

  } catch (evioException *e) {
    cerr << endl << e->toString() << endl << endl;
    exit(EXIT_FAILURE);
  }

  // done
  exit(EXIT_SUCCESS);
}
```

Current Uses

- CODA raw event I/O
- Serialize objects to binary array
  - Transport via cMsg or other protocols (e.g. CLARA)
  - Storage on disk (e.g. DANAEVIO)
- Geant4 output
- Input to event display

Utilities

- Convert from EVIO to XML
  - Then can use XML browser
- Convert from XML to EVIO
  - Doesn't handle string arrays yet…
- Extract, copy events
- Use XML browser after conversion from binary

Future Improvements

- Eliminate event blocking
  - Vastly simplifies I/O code
- Random-access I/O
  - Using in-memory index created when file opened
- Improve in-memory tree query (esp. in Java)
  - Goal is same power in C++ and Java
  - Can implement multiple query models

Summary

- Code sharing is win-win-win
  - CLAS12 – Hall D – JLab/DOE
- Many opportunities for joint code development
  - No-brainer in online
  - Quite feasible in offline
- Many packages already shared
  - Some with joint code development
- EVIO package
  - Easily modified to meet future CLAS12 and Hall D needs