File Output Using RootIOSvc

Brett Viren

Physics Department

Daya Bay US Offline, 2009/07/02

Overview

- I have been developing the low level file I/O services, converters and data objects
- Work is in `dybgaudi/RootIO/` area (not everything uploaded to SVN yet).
- Looked to BesIII/GLAST `RootCnvSvc` for inspiration but all code written from new.
- Initial focus is on getting output working, input not as far along yet.

Currently Existing Packages

In `dybgaudi/RootIO/`:

- **RootIOSvc** Conversion service, converter base classes, low level ROOT I/O streams.
- **TBaseEvent** TObject versions of DataModel's BaseEvent, converters and dictionaries for these objects.
- **RootIOTest** a package providing some drivers for testing.

`RootIOCnvSvc`: top level conversion service

Responsible for:

- Creating a `RootIOAddress` (a `IOpaqueAddress`) based on the `DataObject` being converted.
- Mapping a `RootOutputStream` to a TES path.

It is configured with:

OutputFileMap a mapping of TES path name to ouput filename with special TES path "default" used as catchall.

DefaultOutput another way to specify the catchall, overrides the above.

EventDataService specifies the ES to use.

Converters

Inheritance change for converters:

RootIOBaseCnv provides basic IConverter requirements. Delegates through `DataObjectToTObject()` method.

RootIOTypesCnv templated, manages type conversions between actual and base data objects, creation of output stream and association of object pointer to tree and final writing. Delegates to `DO2T()`.

TemporalDataObjectCnv (example concrete converter). Implements `DO2T()` with minimal code to copy from `DataObject` version to `TObject` version.

RootIOAddress

- RootIO's `IOpaqueAddress`
- Encapsulate any information needed to write out an object:

  filename output filename

  tespath TES path to DataObject and TDirectory path to TTree/TBranch.

  entry entry number (for input)
- Associated with each DataObject in the TES
- Need to understand how it will be used with AES

Low Level `RootOutputStream`

- Abstracts file output to a single "stream" of data.
- Stream is one `TTree` in one `TFile`.
- Stream can be broken into multiple output files (eg, for subruns, or limiting file size).
- Streams independent, can put different trees in different files.

Current Status

- BaseEvent objects written
- Their converters written and working
- Low level RootOutputStream tested and works.
  - Caveat: must use rootcint, not genreflex to produce dictionaries of TObject based classes.

Next Steps

- Move on to input side of things:
  - Low level RootInputStream works and already tested (see next)
  - I think I understand how RootCnvSvc does input and will emulate that.
- Meet up with Simon's work on the RegistrationSequence.
- Fill out needed TObject versions of all DataModel classes and converters
  - See if G.O.D. can help auto-generate either or both

$\longrightarrow$ File I/O done.

Sneak peak: `RootInputStream`

- Abstract the reading of input from file into a "stream" of data.
- Forward, backward and random access based on global entry number.
- Can chain multiple files together, navigation spans files seamlessly.
- Multiple trees in the same file handled as separate independent streams. Ditto trees in different files.