Trybos and Bank Class Support

Rob Kennedy

05-Feb-1997

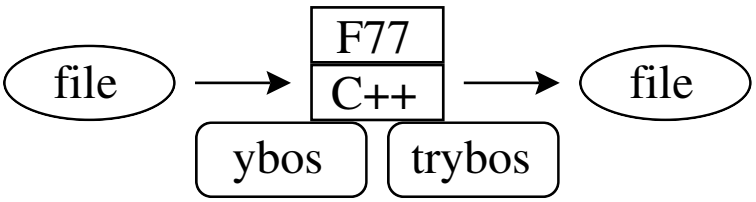
1) <u>Current Release Status</u>

a) /cdf/upgrade: V03-00-09 (current), V03-01-17 (development)
b) cdfsoft2: V03-00-09 (current), V03-01-18 (CVS head)

2) <u>Design Goals for Trybos Version 3</u>

(Implemented *now* in V03-01-18 or in V03-01-19 to be released)

a) Enable A_C++ to use F77/YBOS and C++/Trybos modules



b) Conform to FNAL style and cdfsoft2 s/w mgmt guidelines
c) Add methods to enable Bank Class support tools develop.

(Will implement in V03-02-nn, V03-03-nn, ...)

e) Overhauled doc of product and separately of intended use
f) Safe, polymorphic iterators; split into any-name, same-name
g) Bank lists for I/O, changes to prepare Trybos for F77 use
h) Bank Class Support (partial at first, maturing as proven)

3) <u>Bank Class Support Strategy and Tools (*moved up in priority*)</u>

a) Provide implementation classes for common data formats
b) Extract type info from banks in data stream, feed into DB
c) From DB or header, create a Bank Class w/ generic accessors
d) Imbed hooks in header files to allow revision of C++ details

Presentation to the CDF Computing Upgrade Meeting

Outline of Current Strategy

The Situation

*) Many existing banks with incomplete, chaotic F77 descriptions... automated bank class generation from F77 includes not useful

*) Goal: F77 programmers work in same environment as Run I. As such, no explicit IDL-based approach to support F77/C++ banks.

*) Trybos is likely to evolve, with minor changes needed to all bank classes in the non-accessor section of class (ctor, dtor, assign, ...)

Current Strategy (not new)

1) A number of data access utility classes will be developed to treat common data formats. These can then be privately inherited by banks classes to re-use to data access algorithms.
Examples: fixed-size mono-type <class Type>, fixed size mixed type, blocked D-style with index, blocked D-style with iterator access, ...

2) Convention on implementations: bank NAME in Banks product

```
class header          = NAME_Bank.hh          (includes inlines)
data layout header = NAME_Bank_format.hh (still needed?)
class source          = NAME_Bank.cc          (optional)
```

*** Policy? ***: However we store floating point quantities in banks, do we access and work with quantities outside of banks as double precision? Calculations in double precision are as fast or faster than single precision on all supported platforms

Outline of Current Strategy (cont.)

Current Strategy (new)

3) To initialize large-scale development of bank classes, I can access the type information stored in banks and convert to human-readable string (in the format used by YBOS's BTYMAK()). Read through a variety of data files, putting bank and type string info into a database.

4) Initial guess: assume banks with non-varying type description are fixed-size banks. From the type string info, an initial bank class header is generated with generic names for the accessors. The accessors can be implemented using the data access utility classes. A user provides some input before treating variable-size banks similarly.

5) Keywords and other information are stored in C++ comments in the bank class headers to identify different sections of the class definition. If changes to the general bank class structure are required, then procedures can be written to read the class headers and make the appropriate changes.

6) A user familiar with the intent of a bank's data format can then provide meaningful names to the data accessors, as well as correct assumptions made to initialize bank class development. These may be stored in the bank type information database, or in C++ comments.

7) Some information, perhaps an IDL-like data description, is stored in a database or in comments. This can be extended to F77 includes.

Note: Bitfields, banks not in IO stream are not treated automatically.

Aggressive Time Goals

Note: Releases are now made to cdfsoft2 area, not to /cdf/upgrade.

07-Feb-1997: V03-01-19 (already prepared for release)
- type2string()

14-Feb-1997: V03-02-“00”
- Minor modifications to help A_C++ support F77 modules
- Polymorphic iterators (not safe yet)
- Separate any-name, same-name iterators
- Initial version of overhauled documentation

21-Feb-1997: V03-02-“01”
- Bank tool to accumulate type information
- Bank tool to produce initial bank class header from type info
- First few data access utility classes
- First few bank classes

28-Feb-1997: V03-02-“02”
- Initial version of all bank tools in place
- Overhauled documentation completed
- Safe iterators, if consistent with enabling F77 use

14-Mar-1997 (?): V03-03-"00"
- Internal restructuring to support YBOS simulation for F77 use
- Bank IO list support, etc.

28-Mar-1997 (?): V04-01-"00"
- Partial YBOS simulation for F77 use