Subject: Re: DLM returning a pointer...
Posted by Craig Markwardt on Mon, 23 Apr 2001 21:01:01 GMT
View Forum Message <> Reply to Message

Randall Skelton <rhskelto@atm.ox.ac.uk> writes:
> Hi all,
>
> I am trying to write a few IDL functions which mirror those of a C library
> I frequently use for getting data directly from a unix database
> connection. My problem is that the interface document states that the
> internal C structures should not be directly used; rather, an additional
> layer of abstraction should be used. The basic operation that I want in
> IDL is the ability to (1) connect to the database, (2) give a string of
> queries, (3) get a stream of data, and (4) close the connection.
> Ideally, I would like each of these steps to be a separate IDL function
> that mirrors the C functions. Implementing step 1 is proving to be
> somewhat difficult with the abstraction requirement. In order to establish
> a connection I use a C function:
>
> DBcon *DBconSet(char *host, char *port, char *dbName, ...)
>
> which returns DBconn (a nasty structure that, for the reasons above, I
> don't want to emulate directly in IDL). Nevertheless, this structure is
> passed in subsequent functions so I need the structure to proceed with
> steps 2, 3, and 4.
... deleted ...

I don't think it would be wise to return a pointer, although
technically it is possible. You could in principle cast the pointer
to an integer, and return the integer. Then, in the subsequent calls,
you would cast back to a pointer.

However, this is pretty dangerous since there is always a risk that
the pointer gets corrupted while in IDL-land (for example, what if you
set it to zero? Worse yet, what if it gets to some random part of
memory which then gets overwritten).

A better choice I think is to have your interface routine internally
maintain a *list* of DBconn connections. Then you can have the
interface for DBconSet return an integer index into this list.
Subsequent calls to the other library routines then simply have to
look up the pointer in the list.

Error checking here is easier and safer as well, since you can verify
that the index is GE 0 and LT N_PTR, and you can be sure that only
your wrapper is modifying pointer values in the list. All you need is
a little bookkeeping.

Good luck,
Craig

--

---------- ----------
Craig B. Markwardt, Ph.D. EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
---------- ----------