Subject: Re: DLM returning a pointer...
Posted by Craig Markwardt on Tue, 24 Apr 2001 19:06:14 GMT
View Forum Message <> Reply to Message

Randall Skelton <rhskelto@atm.ox.ac.uk> writes:
> After reading Craig's email, I am somewhat confused...
>
> What I really need to know is how do I allocate a memory block in my C
> function such that I am sure that it cannot be overwritten or otherwise
> corrupted. I want a routine that can establish a connection to a given
> 'port' which requires me to allocate some memory and return a pointer to
> this memory block. I then want to be very sure that the memory allocated
> is 'safe' while now that I am back in IDL. Then I want to be able to pass
> this pointer from IDL back to C so as the database connection must be
> established before data can be sent or received. Finally, from IDL I
> would close the connection (again requiring me to pass this pointer) and
> de-allocate the memory. Obviously, such an implementation is risky as it
> could lead to memory leaks in IDL if the programmer fails to close the
> connection properly. I am open to other ideas, but I want to separate the
> open and close connection functions. I am thinking about putting a
> time-out on the connection so that if idle for more than n minutes it
> deallocates. I fear, however, that a time-out would likely lead to
> problems and would be rather tricky to implement. It would be nice if
> there were some way to ensure that there was a matching 'open' and 'close'
> connection function with the IDL compiler...

Randall, if I'm not mistaken, it's the job of the DBconset function to allocate the memory for the DBcon structure, so in principle you don't have to worry about that. What I am suggesting is to maintain a list of such pointers in your C wrapper. So it could be something like the below. Here I have made a static list of 10 DBcon pointers, which can hold up to ten simultaneous connections. If you want to get smarter, you could dynamically allocate memory if the number of simultaneous connections is unknown and > 10.

Of course I haven't done any error checking, and the dealing with IDL <-> C translation isn't handled here. I'm just showing how a C pointer can be converted to and from an integer value using a lookup table (the table is dbcons[] in this case).

The question of allocating memory is really a separate question altogether. My personal feeling is that there is no harm to use malloc(), but you can use the IDL_MemAlloc() too. Both of these are for raw memory. The other functions are for allocating IDL variables, which carry along other baggage which you don't need here, and as you say, IDL can trample on that data. Of course it will persist across function calls. [If your DLM is unloaded with .IDL_SUPER_SESSION_RESET or whatever I have no idea what happens.]

Good luck,
Craig

```
#define DBMAXCON 10
/* Global lookup table - default initialized to zero */
static DBcon *dbcons[DBMAXCON];

int dbconset_wrapper(char *host, char *port, char *dbname, ...)
{
  int i;
  DBcon *dbcon;

  /* Search for an available slot - one with a zero */
  for (i=0; i<DBMAXCON; i++)
    if (dbcons[i] == 0) break;
  if (i == DBMAXCON) { Maximum connections reached! }

  /* Open the connection and return */
  dbcon = DBconSet(host, port, dbname, ...);
  if (dbcon == 0) { report error! }

  dbcons[i] = dbcon;
  return i;
}

int dbclose_wrapper(int num)
{
  /* Error checking */
  if (i < 0 || i >= DBMAXCON || dbcons[i] == 0) { Invalid handle! }

  DBclose(dbcons[i]);
  dbcons[i] = 0;      /* Reset to zero so it can be reused. */
  return 0;
}


DBcon *DBconSet(char *host, char *port, char *dbName, ...)
```

--

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