Subject: Re: DLM returning a pointer...
Posted by Richard Younger on Tue, 24 Apr 2001 14:34:53 GMT
View Forum Message <> Reply to Message

Randall Skelton wrote:
>
> 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...
>
> Will the memory allocated with the IDL_GetScratch function span the forked
> C process life? i.e. if I use IDL_GetScratch to allocate memory, will IDL
> (potentially) cleanup and deallocate the memory before I call
> IDL_Deltmp()? What about IDL_MemAlloc and IDL_MemFree? Should I just
> consider defining an list say 10 of these structures with IDL_MemAllocPerm
> (giving me 10 possible connections) and forget about reclaiming the
> memory?
>
> (I am assuming here that since IDL is calling the C program, this is a
> unix fork process giving C access to IDL's memory space alone. I am
> reluctant to use malloc directly in C as I doubt that IDL would respect
> the memory it allocates when I return to IDL).
>
> All comments, suggestions and queries are greatly appreciated!
>
> Randall Skelton
>
> NB: just wait until I start asking how to make this multi threaded with
> asynchronous output from simultaneous connections ;)
>
> On Tue, 24 Apr 2001, Martin Schultz wrote:

>
>>>> Hi all,
>>>>
>>>> I am trying to write a few IDL functions which mirror those of a C library
>>>> [...]
>>>
>>> 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.
>>
>> ... you probably meant an unsigned 64-bit value (in IDL speak
>> ULONG64).
>>
>> Martin

Hi, Randall.

I've had a bit of a similar problem in a data acquisition routine. Fortunately (for me) the 3rd-party drivers I'm using don't require me to keep hunks of memory, just pointers out of IDL-space, so I can't speak too knowledgeably about the memory allocation issues.

But one thing to consider might be to use an object to store your pointers. Your object would have simple members to call your C functions. The size of unsigned IDL variable to use is probably best matched to your C pointer size, to 32- or 64- bit. Depends on your platform, I'd guess.

Here comes the good part. You can use the init and cleanup routines to enforce the off main level IDL allocation / deallocation requirement, and cleanup and deallocation should be done whenever the object is destroyed by IDL. You can also create as many connections as you like without having to resort to global memory, or worrying about some connections stepping on others. I'd think you could use IDL_MemAlloc() to get the memory, since you're more or less assured to deallocate it later, but I'm not positive.

Rich

--
Richard Younger MIT Lincoln Laboratory
Email: younger@ll.mit.edu Mail Stop C-130
Phone: (781)981-4464 244 Wood St.
Fax: (781)981-0122 Lexington, MA 02144-9108