Subject: Re: yet another idl memory question
Posted by Karl[1] on Thu, 04 Jun 2009 20:06:28 GMT
View Forum Message <> Reply to Message

On Jun 4, 11:06 am, David Fanning <n...@dfanning.com> wrote:
> Jean H. writes:
>> in which case you may want to save the
>> variable(s) to a file, free the memory (data=0B) and start processing
>> the next file.
>
> Of course, data=0B doesn't free *all* the memory,
> and doing this many times leads, I suspect, to the memory
> fragmentation that is the heart of the problem. I suggest
> you use UNDEFINE. That really does release *all* the memory
> associated with a variable. And it elegantly indicates
> what the code is actually doing, too, a significant
> advantage for people reading your code after you have
> run off to the tropics with that hot financial analyst
> over in the head shed. :-)
>
> http://www.dfanning.com/programs/undefine.pro
>
> Cheers,
>
> David
>
> --
> David Fanning, Ph.D.
> Coyote's Guide to IDL Programming (www.dfanning.com)
> Sepore ma de ni thui. ("Perhaps thou speakest truth.")

I'm not sure that's the explanation. IDL variables consist of a small (about 40 bytes) structure. If the variable has a dynamic part, as in arrays, the dynamic part is allocated separately and the pointer placed in the IDL variable structure. data=0B just frees the dynamic part. Further, I think that the variable structures are allocated in groups (pools) so that IDL does not have to constantly alloc and free these small structures as variables activate and deactivate. So, an undefine will probably just put the variable back into a pool, with no storage impacts at all.

Fragmentation could be the issue, if the right pattern of dynamic allocations and frees exist. But given that this is on 64-bit Linux, I rather doubt it, as the virtual address space is huge and would be difficult to fragment with 750MB blocks in "a few iterations".

To the OP:

Who is actually allocating the memory? OpenDAP? Or your IDL application?

As mentioned before, be sure you are freeing the memory before the next iteration. Regardless of who allocates it, leaving it unfreed will probably bump you up against your process size limit in Linux.

Does OpenDAP allow you to pass in a variable that you have defined/ allocated yourself? You seem to imply that that this is possible since you mention creating your own pointer vars, etc. Are you sure that OpenDAP is actually using the block you pass in and not allocating a new one anyway?

I think you are on the right track for the ideal solution. You want to determine the required size and allocate the block once. Then convince OpenDAP to use it to store the data, and then just keep reusing it. You are fortunate here in that the data size does not change. If it were variable, you'd either have to alloc a worst-case block and live with that or possibly suffer fragmentation issues by allocing/freeing each time.

I think that the key to the answer is determining how OpenDAP allocs this memory and how to convince it to reuse the same block. And you'd need to know more about OpenDAP to determine all of that. Sorry, I don't know much about it.

Karl
