Subject: Re: dlm returning ptr array and string array
Posted by Randall Skelton on Mon, 03 Mar 2003 20:00:00 GMT
View Forum Message <> Reply to Message

I'll bite.

> I started a bit in dlm programming. Because of the latest bug in
> implementing the netCDF library I have to think in this case about getting
> more independent from rsi.

Re-implementing the interface to a file format such as netCDF would probably be a fair amount of work.

> Now my questions:
> Is it possible to return a pointer vector by a dlm?

I am not entirely sure I understand what you are asking here. You cannot create an IDL pointer in C, so if that is what you are asking you'll need to rethink things. You can, create and return IDL variables and arrays of any type (other than pointers). Likewise with structures but you cannot directly interface these as objects (unless, of course, Ronn has some new tricks to show us).

You can, however, allocate some very complex hash, linked-list, or other structure that IDL has no knowledge about in C and pass a simple string/index array back to IDL. You could even go as far as simply keeping a hash table of indicies, strings, pointers and/or file offsets that point to your data. Then when you want a specific data block, you would use the name/index/??? and a separate c routine to return the block as a structure or array. This is what diploma/co-op students are for ;)

> Could they freed by idl?

Again, there is no way to create or act on an IDL pointer in C. You can create an return your large structure in C and then index it in IDL but that isn't particularly clean or easy.

> Is this mostly stable?

Passing ordinary variables, arrays and structures is defined by the stable DLM API. So, yes.

> Does it give memory leakage?

Depends how good of a C programmer you are ;)

> Are there memory limitiations generally using dlms?

Not really. Because DLMs are raw C code that are linked to IDL as shared libraries, you basically have access to as much memory as you have available. Once you allocate memory on the C side, you use various components of the API to pass pointers to this memory so IDL knows about it.
