Subject: DLM heap variable access
Posted by penteado on Sat, 27 Jun 2009 19:34:21 GMT
View Forum Message <> Reply to Message

I got tired of waiting for the ITTVIS folks to implement some more data structures in IDL. Coding a bunch of them (lists, maps, stacks) in IDL would be a fair amount of rewriting the wheel, and also inefficient, because of the way IDL' s pointers and scalars work. So I decided that the nicest solution would be to have IDL objects as wrappers to C++ containers. It is simple enough to do it writing a DLM in the way Ronn Kling' s book suggests, with the IDL object containing a (real) pointer to the C++ object, and wrapper methods to call the C+ + methods.

However, I was unhappy with having to make a method in IDL that passes the object pointer and the arguments to a C++ wrapper, that then does the job with the C++ object. It would be much nicer to write the IDL method directly in C++. The trouble is how to get access to the IDL object's self from the C++ routine, to retrieve the C++ object pointer in it. As Ronn mentions, the IDL object reference gets passed to the method in argv[0], but nowehre I could find a reference to how to use it, except for this very unsatisfying sentence

"Direct access to pointer and object reference heap variables (types IDL_TYP_PTR and IDL_TYP._OBJREF, respectively) is not allowed."

from IDL' s documentation. I figured that the IDL object reference is passed in argv[0] for some use, and it appears that some objects written by ITTVIS do exactly that. So after some experimenting and browsing through idl_export.h, I eventually figured out how to do it.

In the description below, the IDL object was defined with a single structure member, self.obj, that is a pointer to a byte array where the C++ pointer is stored (as suggested in Ronn's book).

1) argv[0] has a type IDL_TYP_OBJREF. Therefore, its value contains the heap variable identifier (IDL_HVID hvid). Of course that is just IDL' s id number for the heap variable, not an actual pointer.

2) idl_exports.h contains the prototype:
IDL_HEAP_VPTR IDL_CDECL IDL_HeapVarHashFind(IDL_HVID hash_id)
I found that this function returns a pointer to the heap variable given its identifier.

3) What heap variable is pointed to by argv[0]->value.hvid? The IDL object's self!

4) It is now necessary to retrieve the heap variable pointed to by

self.obj. This is done with IDL_HeapVarHashFind on the heap variable id in self.obj:

```
//Get a pointer to self (self is {pp_stl,obj:ptr_new()}):
IDL_HEAP_VPTR ohvptr=IDL_HeapVarHashFind(argv[0]->value.hvid);
//Get the identifier of the heap variable of self.obj:
IDL_HVID *pind=(IDL_HVID *) ohvptr->var.value.arr->data;
//Get a pointer to *(self.obj):
IDL_HEAP_VPTR hvptr=IDL_HeapVarHashFind(*pind);
//Get the real pointer from *(self.obj):
memcpy(&object,hvptr->var.value.arr->data,sizeof(object));
```