Subject: Re: Pointers in IDL
Posted by Rick Towler on Tue, 13 Apr 2004 18:06:11 GMT
View Forum Message <> Reply to Message

"Benjamin Hornberger" wrote...

> 1. What are null pointers for?

There may be others but the big one is object class definition. I am going to butcher the explanation but...

When you define the named structure of your object class you define the element names and types. When you create an instance of your object pointers and objects in this "self" structure are null, regardless of how you defined them.

So say you wrote your object definition like so, using the /ALLOCATE_HEAP keyword when defining your myPointer:

```
pro myobj__define

   self= {myobj, $
         myPointer:PTR_NEW(/ALLOCATE_HEAP) $
         }
end
```

If, when creating an instance of your object, myPointer was undefined (not null) you should be able to assign it a value by simply dereferencing it:

```
function myobj::init, data

   *self.myPointer = data

   RETURN, 1

end
```

But this would fail with an "Unable to dereference null pointer" error. IDL ignores your /ALLOCATE_HEAP keyword and always assigns null pointers to pointer and object reference variables in class definition structures.

Our simple object would typically be written like:

```
function myobj::init, data

   self.myPointer = PTR_NEW(data)
```

```
   RETURN, 1

end

pro myobj__define

   null = {myobj, $
          myPointer:PTR_NEW() $
         }
end
```

> 2. If I point a pointer to a variable (e.g. *ptr=indgen(100)) and later
> point it to a smaller variable (*ptr=indgen(50)), do I have a memory leak?
> I.e., do I have to free it before I re-reference it?

No. You only "leak" when you forget or are unable to free the pointer when you are finished with it.

So the following is fine:

```
ptr = PTR_NEW(indgen(100))
*ptr = 'String'
*ptr = FINDGEN(50)
PTR_FREE, ptr
```

But this is bad:

```
ptr = PTR_NEW(indgen(100))
ptr = 'String'
```

ptr used to contain the reference to a heap variable, but we lost that reference when set ptr = 'String'. Since we have lost our reference we can't free the pointer and that memory will be lost for the rest of the IDL session. (This isn't entirely true. You can reclaim lost heap variables using the PTR_VALID function. Check the docs.)

> I want to write a GUI which can open files which contain arrays of varying
> size. Is it ok to define a pointer in the GUI to hold these arrays
> (ptr=ptr_new(/allocate_heap)), and then whenever I open a new file, just
> dereference to the new array (*ptr=array)? Or do I have to free the pointer
> when I close one file and open another one?

No need to free until you exit your application.

-Rick