Subject: Re: Passing Image Data :)
Posted by David Fanning on Tue, 23 Oct 2001 20:18:20 GMT
View Forum Message <> Reply to Message

Logan Lindquist (llindqusit@mrdoc.cc) writes:

> Dr(.) Fanning,
> [I thought Dr. had an period after it because it is an abbrevation of
> doctor? I do not know what Andrew Cool is talking about.]

Please, "Hey, Bro" is fine. :-)

>> What you want in your info structure image field is a pointer
>> to the image:
>>
>> info= { image:Ptr_New(myimage), ...}
>
> I am wondering if you could clear up a couple of things about pointers in
> IDL. How come myimage does not have to be defined during initalization? Does
> the statement above create space in memory for a variable of indefinite
> size? It seems to operate this way., where the data in memory is allocated
> once the data has to be stored to the pointer array. Maybe I am
> understanding pointers incorrectly.

Well, it might *seem* to operate this way, and
in fact there is no real harm *thinking* it operates
this way, but the facts are slightly more complicated.
In truth, it operates just like this:

```
IDL> a = reallybigdata
IDL> a = reallylittledata
```

That is to say, pointers use IDL's normal ability
to allocate memory on the fly and change the type
and structure of IDL variables on a whim.

Where you are probably going wrong is thinking that
IDL pointers are anything at all like C pointers. They
are not. In fact, the only thing that is even remotely
similar about them is their name. :-)

Pointers in IDL are simply normal variables that exist in
a globally accessible area of memory that IDL itself
manages. Nothing more or less. The fact that they
are globally accessible gives them properties that
are useful to us. Namely, we can have multiple program
modules accessing large amounts of data just by carrying
around a light-weight token that you can easily think

of as a secret decoder ring. If you know the code, you
can see the data. Simple as that.

In fact, if you know the code, you can change the data
encrypted by the code. IDL itself will take care of
all the messy details of how this is done. (CIA
special ops, is what I hear.)

> 1.. The Pointer is created - a variable that 'points' to space in RAM
> reserved for a variable of indefinate size.

No, it actually points to a variable of a very defined
size: the size of the variable it points to. If you
don't know how big the data is going to be, or it you
don't yet have a variable to store there, but you still
want a valid pointer, you can make the pointer point to
an "undefined" variable. Undefined variables are perfectly
legitimate variables in IDL:

```
IDL> ptr = Ptr_New(/Allocate_Heap)
IDL> HELP, *ptr
   <PtrHeapVar110> UNDEFINED = <Undefined>
IDL> Print, Ptr_Valid(Ptr)
   1
```

> 2.. The data is read into RAM during the read_image.pro.

Yes.

> 3.. The Pointer then needs to store the image data for future reference.
> This is done by '*info.image = newimage'. Where newimage is the image data
> in RAM.

Yes,

```
IDL> *ptr = newimage
```

Now the pointer points to a defined variable that is
a defined size. IDL itself has managed all the memory
allocation, etc. for you.

> 4.. Is the data then copied into the space originally allocated for it or
> does it simply change it's reference so as to point to the location in RAM
> where the image data was read into?

Uh, I don't know. And frankly, I don't even want to know.
It's fast, so I suspect the image data itself doesn't move.
I prefer to think of it being done with smoke and mirrors,

since it seems more mysterious that way. The point is, even
if you knew how it was done there is nothing you can do in
your code to affect how it is done, so there really is no
point in worrying about it.

> I went back and reviewed how pointers are treated in C++. I was wondering if
> I made my Struct a pointer, could I access memebers of Struct's using the
> '->'?

Don't ever review anything in C or C++ if you want
to know about IDL. It will just confuse the daylights
out of you. :-)

Cheers,

David

--
David W. Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155