Subject: Re: Help: Pointer
Posted by VU KHAC Tri on Fri, 02 Apr 1999 08:00:00 GMT
View Forum Message <> Reply to Message

>> In C, I can write:
>>
>> void XXX ()
>> {
>> int C = 4;
>> int *pC;
>> pC = &C; /*pC points to C*/
>> }
>>
>> I cannot find out how to do this in IDL ?
>
> IDL> c = 4
> IDL> pC = Ptr_New(/Allocate_Heap)
> IDL> *pC = c
>

I found this :

```
IDL> c = 4
IDL> pc = PTR_NEW(/ALLOCATE_HEAP)
IDL> *pc = c
IDL> print, *pc
      4
IDL> c = 5
IDL> print, *pc
      4
```

But in C, when c is modified, (*pc) is also modified because pc points
to the memory where c is. That mean, if c = 5, *pc = 5.

Thank you so much for your book.
Best regards,
Tri.

Subject: Re: Help: Pointer
Posted by davidf on Fri, 02 Apr 1999 08:00:00 GMT
View Forum Message <> Reply to Message

VU KHAC Tri (tvk@info.fundp.ac.be) writes:

> In C, I can write:
>
> void XXX ()

> {
> int C = 4;
> int *pC;
> pC = &C; /*pC points to C*/
> }
>
> I cannot find out how to do this in IDL ?

```
IDL> c = 4
IDL> pC = Ptr_New(/Allocate_Heap)
IDL> *pC = c
```

But most people would just to this:

```
IDL> pC = Ptr_New(c)
```

To destroy the pointer (and the thing it points to),
be sure to do this when you are finished:

```
IDL> Pointer_Free, pC
```

If you are writing a widget program you can clean up
your pointers in the CLEANUP procedure you assign
to your top-level base with the XMANAGER command.

Cheers,

David

P.S. People may be interested to know that a book is
winging its way to Tri today. :-)

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

Subject: Re: Help: Pointer
Posted by steinhh on Sat, 03 Apr 1999 08:00:00 GMT
View Forum Message <> Reply to Message

Vu Khac Tri wrote:

> But in C, when c is modified, (*pc) is also modified because
> pc points to the memory where c is. That mean, if c = 5, *pc = 5.

You're right. IDL doesn't have pointers in the same sense
as C does - there is no "address of" operator, and there
is no way to make a pointer that points to a normal variable.

The closest thing (and it's close enough to emulate
the C behaviour, but not it's notation) is:

```
IDL> c=ptr_new(4)
IDL> pc=c
IDL> print,*pc
      4
IDL> *c = 5
IDL> print,*pc
      5
```

I.e., you have two pointers, pointing to the same heap
variable. The only difference from C notation is that
you have to dereference *both* variables as pointers.

Regards,

Stein Vidar
