Subject: Re: Pointers in IDL
Posted by R.Bauer on Tue, 13 Apr 2004 17:37:02 GMT
View Forum Message <> Reply to Message

Benjamin Hornberger wrote:

> Hi all,
>
> I still don't understand all aspects of pointers in IDL. 2 Questions:
>
> 1. What are null pointers for?

If you have a special list (in german it is verkettete Liste) then it is
easy to recognize the end of the list. Because this is normally a null
pointer. Otherwise it is a pointer to a pointer in the list.
This is often used for search algorithm.
You get a null pointer if you free a pointer by ptr_free too. The status of
a pointer could be tested by ptr_valid(). If it is 1 you could dereference
the pointer otherwise it is a null pointer.

At the end of r program you should free all your pointers. Because pointer
are global variables in case of the memory usage.

I read that they can't be dereferenced.
> What is their purpose then? The Gumley book writes (pg. 61): "Null
> pointers are used when a pointer must be created, but the variable ...
> does not yet exist." What would I do then when the variable does exist
> later and I want the pointer to point to it? Wouldn't I use
> ptr_new(/allocate_heap) in the first place, i.e. not create a null pointer
> but a pointer to an undefined variable? Can anyone give an example when I
> would use ptr_new()?

```
ptr1=ptr_new(5)
a=findgen(10)
ptr2=ptr_new(a,/no_copy)
help,a
ptr3=ptr_new(/allocate_heap)
*ptr3='ALPHA'
```

If you would like to define a complex structure with pointers and you
haven't all data at the moment of the definition I myself define them with
a string constant ('UNDEFINED')
e.g.

```
pi={name:'UNDEFINED','email':'UNDEFINED'}
```

Later on it is easy to replace the value by a new assignment and you have
not to test always if it is a Null Pointer or could be written.
Another reason for this is you can easily check if the value is different
from 'UNDEFINED'. tags with values of 'UNDEFINED' could be removed.

>
> 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?

This makes no difference, because Pointer in idl are totally different from
pointer in C. There is a heap control mechanism in idl which controls all
the pointer assignment and the memory usage of them.

With heap_gc you could do a garbage collection of the heap memory.

There was in the past a discussion about memory leaks. Please have a look
into the google archive
http://groups.google.de/groups?hl=de&lr=&ie=UTF-8&am p;group=comp.lang.idl-pvwave

>
> 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?

I prefer *ptr=array

But you should think about a structure if you use more than one pointer.
This structure could be then a pointer too.

Cheers
Reimar

>
> Thanks for your help,
> Benjamin

--
Forschungszentrum Juelich
email: R.Bauer@fz-juelich.de

http://www.fz-juelich.de/icg/icg-i/
========== ======
a IDL library at ForschungsZentrum Juelich
http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_lib_intro. html