Subject: Re: Storing !NULL in struct
Posted by Yngvar Larsen on Mon, 18 Mar 2013 18:26:51 GMT
View Forum Message <> Reply to Message

On Monday, 18 March 2013 17:48:54 UTC+1, Mike Galloy wrote:
> On 3/18/13 6:40 AM, Tom Grydeland wrote:
>

>
> I think what you are showing here is that any variable passed by value
> does not end up modified at the calling level.

Right.

> It doesn't work for arrays of structures either:
>
> IDL> sarr = replicate({ foo: 0 }, 10)
> IDL> modify, sarr[0]
> IDL> print, sarr[0]
> {   0}

Yes, but you can still modify an element, or even a range of elements, of a structure array at your current calling level:

```
IDL> sarr[0].foo = 4
IDL> print, sarr[0].foo
      4
IDL> sarr[3:5].foo = 4
IDL> print, sarr[0:6].foo
      4       0       0       4       4       4       0
```

This is not the case for lists, as already discussed:

```
IDL> larr = list(length=10) & for n=0, 9 do larr[n] = {foo: 0}
IDL> print, larr[0].foo
      0
IDL> larr[0].foo = 4
% Attempt to store into an expression: Structure reference.
% Execution halted at: $MAIN$
```

I fail to see why the latter isn't allowed. This means that if you still want to modify structures within your list, you need to do something like this

```
IDL> tmp = larr[0]
IDL> tmp.foo = 4
IDL> larr[0] = tmp
IDL> print, larr[0].foo
      4
```

This seems silly to me. What is the purpose of not allowing this?

Switching to hash tables instead of structures as general purpose data structure is of course a very good strategy in 2013, but I have at least 10 years worth of legacy code which already heavily (mis-)uses the anonymous structure as a "hash table light" with case insensitive keys.

--
Yngvar