Subject: Re: Storing !NULL in struct
Posted by Bob[4] on Tue, 19 Mar 2013 03:26:50 GMT
View Forum Message <> Reply to Message

On Monday, March 18, 2013 12:26:51 PM UTC-6, Yngvar Larsen wrote:
> 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.

I agree. LISTs and HASHes should return a reference to their elements and not a temporary copy.

In addition, IDL needs a reference type that would be similar to a PTR but would not need to be de-referenced to get at what it is pointing at. It could use de-referencing (or a function call) to set the reference but then would allow syntax like a normal variable. This would greatly simplify IDL programing and could perhaps be used to fix the mess that IDL LISTs and HASHes are.
