Subject: Re: Pass by value and performance
Posted by JD Smith on Thu, 15 Dec 2005 21:04:18 GMT
View Forum Message <> Reply to Message

On Thu, 15 Dec 2005 07:56:50 +0100, Antonio Santiago wrote:

> Kenneth P. Bowman wrote:
>> Perhaps someone can clarify this for me.
>>
>> I was doing this
>>
>> data = {values : FLTARR(...), $
>> other : other stuff ...}
>>
>> Then pass "data" to a procedure and do this
>>
>> result = INTERPOLATE(data.values, x, y, z)
>>

> I like to understand pointers in IDL in this way:
>
> 1.- 'a' is a conventional variable managed by IDL and its "garbage
> collector".

Sadly, IDL doesn't have garbage collection. It would be nice if it did, but until then, it's up to you to free all of your heap variables at the correct time (which is great when you know when that is).

> 2.- '*a' is a HEAP variable, where 'a' stores a reference to it. Also, the
> content of the variable 'a' is stored in the heap memory.
>
> Then 'a' is a reference for a "normal" variable that stores a reference,
> and '*a' is a reference to a HEAP variable that stores a 5.

I'd just say both a and *a are variables. One ordinary (local in scope), the other heap (global in scope).

> junk, *a --> The content of the HEAP memory variable is pased by value.

This isn't correct. De-referenced pointer variables (aka "heap" variables) are passed by reference, just like regular variables (which they are, really). E.g. in Ken's original example:

result = INTERPOLATE(*data.array, x, y, z) ; by reference

would indeed pass the pointer heap variable by reference and not by value. As such it would be much faster (for large arrays) than

INTERPOLATE(data.array,x,y,z), which would require copying the full array to a local variable, and would be equivalent to a simple INTERPOLATE(array,x,y,z).

As pointed out in the pointer tutorial (http://www.dfanning.com/misc_tips/pointers.html), there is no difference between pointer heap variables and ordinary variables, except in how you access them. Of course, that also means that a structure member (or array element, etc.) of a dereferenced pointer variable is (just like a member of an ordinary variable), still passed by value:

```
 result = INTERPOLATE((*data).array, x, y, z) ; by value
```

Here `data' is a pointer to a structure with member "array", which is passed here by value.

This equivalence also means that standard IDL variable tricks, like re-assigning the memory contents of one variable to another without copying, work just fine for pointer heap variables (and in between plain old variables and pointer heap variables).

JD
