Subject: Re: Reading Values into Individual Array Elements
Posted by Ken Knighton on Mon, 15 Apr 1996 07:00:00 GMT
View Forum Message <> Reply to Message

Rolando Raqueno <rvrpci@cis.rit.edu> wrote:
> steinhh@amon.uio.no (Stein Vidar Hagfors Haugan) wrote:
>>
>> Since IDL passes subscripted array elements by value and not by reference,
>> a(3) is effectively a constant to READ, because READ cannot change
>> the value of a(3). What you are saying to IDL is therefore basically the
>> same as:
>>
>> IDL> read,"TEST"
>>
>> which is supposed to show the behaviour you described.
>>
>
>
> Rather odd isn't it since IDL is FORTRAN based and passes everything
> by reference

IDL is written in C. C passes everything by value, but if you pass a pointer by value and then use the pointer value to reference memory, you effectively pass by reference. Of course, the argument passing convention of the implementation language used to write IDL is of little interest since IDL has its own argument passing convention. This convention is that everything is passed by reference except for expressions. Expressions include: Literals, function evaluations, arithmetic expressions, subarrays, and structure tags.

Although this argument calling convention may seem strange at first, it makes sense in the context of the rest of IDL. In IDL, any named variable can be changed from one type to another at any time by a simple assignment statement or as the result of a procedure changing one of its passed arguments internally and passing the result back to the caller. As a result, what would happen if you had the following if pass by reference was used?

```
PRO test, a

  a='xyz'

END

IDL> a=[1, 2]
IDL> test, a(1)
```

Either IDL can do type and dimension checking, or some convention has to be used to resolve conflicts like this. I personally think that

it would be valuable to be able to lock a variable's data type and dimension when desired. This would then generate a run time error when code attempted to change the variable's type during an assignment or function call. Of course, this extra checking would slow things down a little bit.

Ken Knighton knighton@cts.com
San Diego CA
