Subject: Re: Reading in to array elements from a file.
Posted by davidf on Wed, 22 Jul 1998 07:00:00 GMT
View Forum Message <> Reply to Message

Neil Rogers (nrogers@dera.gov.uk) writes:

> I wonder if anyone could solve the following mystery for me.
> I am trying to read in *individual* array elements from a file using
> the readf procedure. I have an input file called test.txt containing
> (say)
> 1 2 3 4 5 6
> and the following IDL procedure to read them in and print them out
> again.
> ;------------------
> pro test
> array=intarr(6)
> infile='test.txt'
> openr,inlun,infile,/get_lun
> for cnt=0,5 do begin
> readf,inlun,array[cnt]
> endfor
> close,inlun
> free_lun,inlun
> print,array
> return
> end
> ;------------------
> Unfortunately, although no compile or run-time errors are encountered,
> nothing is assigned to the array elements and the file pointer stays
> at the beginning of the file. The output is then,
> 0 0 0 0 0 0
> Why is this?

The reason you can't read into a subscripted array has to
do with the way IDL passes information into and out of
procedures and functions. Variables are passed by
reference, meaning that it is actually the pointer
(a _real_ pointer here) or address that is passed into the
procedure or function. Changes made to the variable inside
the procedure or function are made to the "real thing".
For example, consider this procedure:

```
PRO JUNK, variable
variable = 10
END
```

If I call it like this:

```
thisVar = 5
Junk, thisVar
Print, thisVar
   10
```

We see that thisVar has been changed by the procedure.

But variables are the *only* things that are passed by reference. Everything else (and this includes things like structure defererences, expressions, constants, and--yes--subscripted variables) are passed by *value*. This means that a COPY of the thing itself is passed into the procedure or function.

So, for example, in our little example if we subscript the variable like this then the variable outside the procedure doesn't change:

```
thisVar = 5
Junk, thisVar[0]
Print, thisVar
   5
```

Notice that this code doesn't cause any error messages. In fact it is perfectly valid IDL code to pass a parameter by value and not by reference.

As it happens, this is *exactly* what happens when you do this:

```
array = IntArr(6)
For j=0,5 DO ReadF, 1, array[j]
```

The value array[j] is passed by VALUE, not by REFERENCE and thus it's value outside the ReadF procedure can never change, although internally to the ReadF procedure it is doing exactly what it is suppose to be doing (i.e., reading a value from the file into a *copy* of a subscripted variable).

This behavior is so fundamental to the way IDL works that it is unlikely to change in our lifetime. :-)

Cheers,

David

David Fanning, Ph.D.
Fanning Software Consulting
E-Mail: davidf@dfanning.com
Phone: 970-221-0438
Coyote's Guide to IDL Programming: http://www.dfanning.com/