Subject: Re: filling an empty array
Posted by Peter Mason on Thu, 04 May 2006 05:34:03 GMT
View Forum Message <> Reply to Message

JJMeyers2@gmail.com wrote:

```
> Hello,
>
> I have an empty array that I am trying to fill out with results from a
> conditional statement. Because I do not know the results of the
> condition (and I do not want to use pointers) I make a really big
> array and I fill it. When I compare the results though with the
> expected ones they are not correct (except the first value). Any
> ideas on what the problem might be?
>
> Here is the part of the code:
>
> index_data=intarr(32,1000)
>
> FOR i=0,31 DO BEGIN
> index_data(i)=
> where((dat_x(*,i) GE -0.1)  AND (dat_x(*,i) LE 0.1)  AND $
>          (dat_y(*,i) GE -0.5)  AND (dat_y(*,i) LE 0.5))
> ENDFOR
>
> The result of the 'where' statement is different for each of the i
> cases and when I hard-coded several integers and compared with the
> results of the loop they were different.
>
> Does anyone have any idea what the problem is?
>
> Thank you in advance,
> JJM
```

JJ, sorry if I sound critical but you mustn't build a single padded index array like this, especially when that index array is only partially valid and therefore unuseable without further extraction.   You just mustn't. Don't do it.   The EPG will get you.   There's got to be a better way. Even doing the actual work row-by-row (i.e., create row's index array and then use it) would be better.
Anyway, some discussion on the current setup.

You are indexing a two-dimensional array (INDEX_DATA) with a one-dimensional index (I).
You *can* do this in IDL, but when you do it works like this: IDL treats the array as one-dimensional (in this case an INTARR(32*1000)) and inserts stuff accordingly, starting at the given index (I).   Memory layout becomes important.

Clearly you are getting a *lot* of overwriting here, as I is only increasing
by 1 in each iteration.

You need to do 2D insertion, as in INDEX_DATA[i,0]=...
You might have to use REFORM(..., 1, ?) on the right-hand-side to get it
2-dimensional (with a leading dimension of 1).

It would be much better if INDEX_DATA was an INTARR(1000,32).   Then you
could do simple, efficient 2D insertion with INDEX_DATA[0,i]=..., or even 1D
insertion with INDEX_DATA[i*1000L]=...

You ought to track WHERE's hit count in a separate 1D array as well.   All
those zero values in INDEX_DATA are valid indices and they will probably
give you a headache downstream.   Also, -1 (bad WHERE) is an invalid index
and you would have to look for it.

There's probably a neat solution with HISTOGRAM but I'm not up to it :-)

Cheers
Peter Mason