Subject: Re: behavior of arrays
Posted by Jack Saba on Thu, 20 May 1999 07:00:00 GMT
View Forum Message <> Reply to Message

I must not have made the point clearly.

David Kastrup wrote:

```
>
> Jack Saba <jack@icesat1.gsfc.nasa.gov> writes:
>
>> But more often than not, I WANT the extra dimension to be lost,
>> or at least I want IDL to be willing to ignore it where appropriate.
>> Consider this unrealistic example that nevertheless illustrates a
>> problem that occurs all too often in IDL:
>>
>> IDL> x=findgen(100)
>> IDL> ijk=where(x eq 10)
>> IDL> for i=ijk,99 do print, i
>> % Expression must be a scalar in this context: I.
>> % Execution halted at:  $MAIN$
>>
>> I could have written i=ijk[0],99, or i=REFORM(ijk),99 to avoid the
>> error. But it shouldn't be necessary -- this should be handled
>> transparently.
>
> It is handled transparently.  If you want a scalar, write ijk[0].
> This works even where ijk is *indeed* a scalar.
>
```

This was only an extremely simplified example of the problem.
I don't want to have to write k[0] every time for a scalar, and IDL
returns these degenerate vectors from a number of built-ins.

To me, having to specify [0] means that the difference between
a vector and a scalar is NOT transparent in those cases where (in my
opinion) it should be. That's not the way I expect a 4GL to act.

When I say IDL should handle degenerate dimensions transparently,
I mean that if there are too many dimension of size 1, they
should be ignored, and if extra dimensions of size 1 is needed,
they should be added automatically; array <--> scalar translation
should be automatic if there is only 1 element in the array.

I'm curious about the opinion of the group on this point. Does IDL
function in this regard as most people want and/or expect, or would
the more transparent behavior be preferred? I admit I hadn't thought
in terms of the problem raised by R. Bauer, who needed the second
redundant dimension that had disappeared. Are there other arguments

for or against?

> This is, BTW, about the only way to check for the result of "where" in a
> useful way:
> w = where(x)
> if (w[0] lt 0)

```
w = where(x,count)
if count NE 0...
```

> ...
>
> --
> David Kastrup   Phone: +49-234-700-5570
> Email: dak@neuroinformatik.ruhr-uni-bochum.de   Fax: +49-234-709-4209
> Institut fï¿½r Neuroinformatik, Universitï¿½tsstr. 150, 44780 Bochum, Germany