Subject: behavior of arrays
Posted by R.Bauer on Wed, 19 May 1999 07:00:00 GMT
View Forum Message <> Reply to Message

I have some problems to understand the logic which may be behind the handling of an array
like defined.

Why is the result help, b.d only [10] and not [10,1] ?

This means I will lose the information that's it is / was a 2-dim dataset.

R.Bauer

```
d=reform(findgen(10),10,1)
help,d
;D               FLOAT     = Array[10, 1]

b=create_struct('d',d)
help,b,/str

;** Structure <1348428>, 1 tags, length=40, refs=1:
;   D               FLOAT     Array[10, 1]


help,b.d
; <Expression>    FLOAT     = Array[10]


c=b.d
help,c
; C               FLOAT     = Array[10]
```

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

> 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)

There is, of course, always the count parameter:
w = where(x, count)

if (count GT 0)

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

---

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

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.
>

Fact is, there are too many situations where it is *not* handled
transparently, and there are situations when you want extra dimensions

to stick around. For example, when we planned our 3D model output analysis tool, the first thought was to store all data in 4 dimensional arrays, signifying x,y,z, and time. But as soon as you extract something out of these cubes, you loose one dimension, and with IDL that really means "loose" so that you don't know any longer whether you have an x,z,t array or an x,y,z array. I know one can "think different" and come up with a working mechanism (we did), but still I sometimes regard this "transparency" issue as quite annoying. But I agree, that at least the IF statement should be able to recognize a 1-element array by itself.

> It is handled transparently. If you want a scalar, write ijk[0].
> This works even where ijk is *indeed* a scalar.
>
> 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)
> ...

well, not the *only* way. You can also test
  w = where(x,count)
  if (count eq 0) then ...

regards,
Martin

--

|||||||||||||||\\\\\\\\\\\\\\\----------////////////// //|||||||||||||||
Martin Schultz, DEAS, Harvard University, 29 Oxford St., Pierce 109,
Cambridge, MA 02138 phone (617) 496 8318 fax (617) 495 4551
e-mail mgs@io.harvard.edu web http://www-as/people/staff/mgs/

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

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 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)
...

--
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

---

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

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.

"R.Bauer" wrote:
>

> I have some problems to understand the logic which may be behind the
> handling of an array
> like defined.
>
> Why is the result help, b.d only [10] and not [10,1] ?
>
> This means I will lose the information that's it is / was a 2-dim
> dataset.
>
> R.Bauer
>
> d=reform(findgen(10),10,1)
> help,d
> ;D   FLOAT   = Array[10, 1]
>
> b=create_struct('d',d)
> help,b,/str
>
> ;** Structure <1348428>, 1 tags, length=40, refs=1:
> ;   D   FLOAT   Array[10, 1]
>
> help,b.d
> ; <Expression>   FLOAT   = Array[10]
>
> c=b.d
> help,c
> ; C   FLOAT   = Array[10]

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

Pavel Romashkin wrote:

> I do not quite realize what is the point in this exercise. FLTARR(10,1)
> is an array with 10 columns and 1 row (which all is meaningful only in
> the direction of array operations which in IDL is row-wise). Therefore,
> seems to me that FLTARR(10, 1) is the same as FLTARR(10) to begin with
> (unlike FLTARR(1, 10)). If you anticipate expanding the matrix, its easy
> to do by using TRANSPOSE and subscripts. I have never experienced a
> problem with losing or mixing up dimensions, although most of my data
> are matrices. In the example below, I see no loss of information in the
> transition from d to c. If you defined more than 1 row, C would be 2D
> in the last statement.
> Cheers,
> Pavel
>

The problem I have is
if I like to write a field (10,1) to a netCDF File only one dimension is
defined
instead of two because size returns only one dimension.

At the moment I have to define _fillValue in the second row to get two
dimensions.

R.Bauer

Subject: Re: behavior of arrays
Posted by thompson on Fri, 21 May 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Jack Saba <jack@icesat1.gsfc.nasa.gov> writes:

> 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?

We've also been bit by the disappearing dimensions problem, where arrays of
dimensions [10,1] (for example) automatically turned into an array of dimension
[10] when we didn't want it to. There are a lot of times when you want this to
happen, but there are also times when you don't. I certainly wouldn't want IDL
to start second guessing us even more than it does already.

William Thompson

Subject: Re: behavior of arrays
Posted by Mark Fardal on Fri, 21 May 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Pavel wrote
> Therefore, seems to me that FLTARR(10, 1) is the same as FLTARR(10) to
> begin with (unlike FLTARR(1, 10))... In the example below, I see no
> loss of information in the transition from d to c...

"The difference between the right word and the almost right word is really
a large matter. It is the difference between the lightning-bug and the
lightning." - Twain

The difference between the right array and the almost right array is unfortunately also a large matter. You may think that no information is being lost by dropping the final dimension, but the dimensionality of the array is itself information. This can mess up matrix multiplications. Herr Bauer says it can also mess up writing to a netCDF file. Who knows where else it could matter.

For a concrete example of the first problem, a couple months ago I posted a way to make CURVEFIT crash. All you have to do is make a single-parameter fit and mix the type (float/double) of the parameters in a particular way. So this problem is biting RSI's own programmers, not just morons like me.

This behavior occurs in part because type promotion routines sometimes (not always) change the dimensionality of the promoted variable, and it's awfully hard to anticipate all the situations where a type promotion could occur. Consider what would happen if the behavior was reversed, and REFORM sometimes decided to change the type as well as the dimensionality of a variable. Would that be okay too? After all, every integer can be represented as a float...

cheers,
Mark Fardal
UMass
