Subject: Re: Problem array subscripting
Posted by Henry Chapman on Wed, 18 Aug 1999 07:00:00 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
>> IDL> result = array[x, y, index, 0]
>
> The trouble with this trick is that I don't understand
> how it works. It's fine for production code, don't get
> me wrong, where a little smoke and mirrors can even be
> elegant. I just don't want to get up in front of an IDL
> programming class and have to explain it. Do you have
> any theories?

Well, here's what's written in "Building IDL applications" on pp 62,63:

---- [begin quote from building.pdf] ---

"Extra" Dimensions

When creating arrays, IDL eliminates all size 1, or "degenerate",
trailing
dimensions. Thus, the statements

A = INTARR(10, 1)
HELP, A

print the following:

A INT = Array(10)

This removal of superfluous dimensions is usually convenient, but it can cause problems when attempting to write fully general procedures and functions. Therefore, IDL allows you to specify "extra" dimensions for an array as long as the extra dimensions are all zero. For example, consider a vector defined as follows:

ARR = INDGEN(10)

The following are all valid references to the sixth element of ARR:

X = ARR[5]
X = ARR[5, 0]
X = ARR[5, 0, 0, *, 0]

Thus, the automatic removal of degenerate trailing dimensions does not cause problems for routines that attempt to access the resulting array.

------- [end quote] ---

So, IDL says that it's all Kosher to put in the extra dimension.

But, I guess, that doesn't quite answer all of it, since the above doesn't address the behaviour of subscripting. Let me quote some more from building.pdf (p 68):

---- [start quote]

When combining two subscript arrays, each element of the first array is combined with the corresponding element of the other subscript array. The two
subscript arrays must have the same number of elements. The resulting subscript
array has the same number of elements as its constituents. For example, the
expression A[[1, 3], [5, 9]] yields the elements A[1,5] and A[3,9].

---- [end quote]

That is, when you try and do

result = array[x, y]

IDL wants to return result[0] = array[x[0], y[0]], ...
result[i] = array[x[i], y[i]], ...

And this may well be what you want. So how does one switch between these behaviours? There is a hint on p. 67 of the same good book:

-- [start quote]

When combining an array subscript with a subscript range, the result is an array of subscripts constructed by combining each element of the subscript array with each member of the subscript range. Combining an n-element array with an m-element subscript range yields an nm-element subscript. Each dimension of the result is equal to the number of elements in the corresponding subscript array or range.

For example, the expression A[[1, 3, 5], 7:9] is a nine-element, 3 ï¿½x 3 array composed of the following elements:

-- [end quote]

So, I think my trick is consistant with IDL's documentation, although it might not have been what IDL had in mind. Here is my summary of array

subscripting:

There are two behaviours you may want when you try result = array[x, y,..., z], and x, y, ..., z are 1-d arrays:

1. return a 1-d array with result[i] = array[x[i], y[i],..., z[i]]
2. return a 2-d array with result[i, j,..., k] = array[x[i], y[j],...,z[k]]

IDL does (1), which requires x, y,...,z all have the same length. However, IDL does (2) if one of the dimensions is a subscript range. Since a single number is a subscript range, it will do (2) in that case. And since 0, written in as an extra dimension is also a subscript range, IDL will use behaviour (2)

Henry.

--

Henry Chapman mailto:chapman9@llnl.gov
Information Science & Technology Program phone:(925)423-1580
Lawrence Livermore National Lab fax:(925)423-1488
L-395, 7000 East Ave., Livermore CA 94550