Subject: Re: Generalisation of the use of lists in the IDL language
Posted by kagoldberg on Fri, 23 May 2014 04:42:56 GMT
View Forum Message <> Reply to Message

I understand that what you're discussing here seems to be the limitations of lists, but I would suggest that you may not be using the right tool for the job. If you've got a list of scalar values, go ahead and use an array. The list doesn't give you any advantage. Lists are like pointer arrays. They're cumbersome for tasks that arrays perform natively, but powerful for mixed datasets, or individual large elements.

Use lists for more complex or varied information. You can have a list of structures, where not every structure is the same. Or a list of images, where each image can be a different size. Those are things you can't easily do with arrays. In the past, you could make a pointer array, but lists just made that a whole lot easier, because of the lack of pointer dereferencing, and the indexing.

Personally, I like the feature where the list's indices can seamlessly burrow into the individual elements. Like this:

```
IDL> a = list([10,11,12,13], [100,101,102,103], ['a','b','c','d'], ['Bob','Tom'])
IDL> print, a[3,1]
Tom
IDL> print, a[1,2]
    102
```

I don't know that this configuration would ever occur quite like this in a program I write, but that's a flexible way to retrieve information. It might be too much to ask to start performing all of the varied array operations on the elements of such a list.