Subject: Re: IDL 8.0 compile_opt changes
Posted by Yngvar Larsen on Wed, 06 Jan 2010 14:56:08 GMT
View Forum Message <> Reply to Message

My 2 cents:

*) About negative indexing:

> For example:
>
> a[0:-1] In IDL80, returns all the elements
> a[-1] In IDL80, returns the last element

Is the only use for negative indexing a way to indicate the last element? Then this is probably the most idiotic choice possible: semantic overload of the integer -1 with no gain whatsoever (except possibly in ITTVis' parsing code that might be simplified by some clever tricks?). Why not use some other syntactic element like the string 'end' (like matlab) or 'last' or '%' or whatever?

Or is there some other clever use? Reverse/step indexing like matlab's "array(first:step:last)" with negative steps allowed would be fantastic, but this doesn't require negative indices at all. I propose:
IDL8> array = indgen(5)
IDL8> print, array[%:1:-2] ; array[first:last:step] like the FOR loop indexing
4 2
IDL8> print, array[%]
4
IDL8> print, array[%/2]
2

etc

*) about '.' instead of '->':

NO,NO,NO!

IDL is not Python, as some people have pointed out. Any syntax that requires access to the dynamic runtime to be resolved is a Bad Idea, which is why the transition from () to [] was a Good Idea. I agree that '.' looks better than '->', but unless access to the internal 'self' structure is allowed from outside of objects, like in Python, I see absolutely no point in looking superficially like Python. That would in fact be _more_ confusing since it is not the same. Semantics is more important than syntax at the end of the day.

*) about legacy libraries:

> 2. Do you use existing libraries (like Astrolib or JHUAPL's) that
> would break? Could ITTVIS retrofit these 2 libraries and give them
> back to the community?

That would be the best solution to avoid scaring the many(?) users of these libraries from upgrading. And how about a tool in the IDE to convert legacy code from old to new style, i.e. () to [] and -> to '.' if this is chosen, optionally interactively. Also maybe including indicating common pitfalls like formatted IO for manual inspection. Should be easy to get 99% right as many already commented on.

*) about default 'idl2' breaking code:

In my opinion, code that is broken by this (mostly formatted IO for binary files of a specific format without specifying explicit datatypes) is in fact already broken by design. Or at least badly written.

I vote for default 'idl2'. 13 years of transition should be sufficient...

*) about a new extension for >=8.0 code: I hate the idea, though it solves the problem. I like better the idea of using the comment character e.g. ;%compile_opt idl1. It is butt ugly, but so is the code that needs it :)

*) about other issues for 8.0 (according to http://michaelgalloy.com/2009/12/28/agu-idl-users-group-meet ing.html#more-2092):

- operator overloading: nice!!
- null elements for arrays and structures: nice!! ('[]' and '{}'?)
- FOREACH loop: very useful. Even better if new data types like maps/lists/hashtables are planned.

Wishlist:
- More general datatypes like map/list/hashtable/tuple etc.
- I wish a scalar and an array with one element could be treated as different datatypes. Mathematically, they are different, so sometimes confusing bugs are caused by conflating them like IDL currently does.
- object based widgets anyone? And when will the new widgets used for the Workbench be available for users? Motif looks so last millenium...
- some kind of package system or namespaces to avoid having to call my own library functions/procedures/objects something like " reallylongstringtominimizeprobabilityfornameclashes_myfuncti on "

--