Subject: New Object Method Invocation Syntax Brokenness
Posted by JDS on Mon, 16 May 2011 17:27:30 GMT
View Forum Message <> Reply to Message

A few years back, we had long discussions about the trouble the new "dot" syntax for method invocation would bring about. I've just stumbled across a new and unexpected case. Not only does IDL 8 interpret "." as equivalent to "->", in certain uses it also goes the OTHER WAY, recasting "->" as "." and overriding a method invocation with structure variable dereferencing, *even when the arrow operator is used*. This is a major issue for any pre-existing class which contains a method-function and structure member with identical names, which is quite common.

Consider this simple class:

```
pro DOT_ARROW::try
  ;compile_opt idl2
  print,"GOT: ",self->item([1,2,3,4])
end

function DOT_ARROW::item, p
  return, size(p,/DIMENSIONS)
end

pro dot_arrow__define
  st={DOT_ARROW,$
      item:0.0}
end

IDL> d=obj_new('dot_arrow')
IDL> d->try
GOT:       0.00000      0.00000      0.00000      0.00000   ;; RUNS without ERROR, but is
WRONG!!!
```

Now with IDL2 enabled (or in IDL 7):

```
IDL> d->try
GOT:            4  ;; RIGHT!!!
```

IDL 8 has gone one step further than introducing a new ambiguity between "." and "->" in the name of Python/JS-esque cosmetics, it's completed that ambiguity -- essentially enforcing it on us -- by making it work both ways. In essence, IDL8 is interpreting:

```
  self->item([1,2,3,4])
```

as

```
  self.item[[1,2,3,4]]
```

(no indexing error since you can't go out of bounds with an indexing vector by default). If I say

"->", I mean invoke a method, no matter what. However, as new programmers adopt "." for method invocation, this ambiguity will never go away.

And this is not a simple matter of troublesome syntax errors. What's really scary about this is, there are plenty of imaginable cases in which a->b(c) is interpreted by IDL8 in a completely different way from earlier versions of IDL, yet it can run through without error, silently corrupting your output. There is no warning against or detection of the use of identical names for a method-function and a class structure variable.

Chris Torrence noticed this ambiguity could cause trouble while IDL 8 was being designed; I don't recall what the final decision was. But in any case, my understanding was this would be a new ambiguity, affecting new code which uses the dot operator for method invocation. Never was it discussed that IDL 8 would render inoperable (or worse, operable but incorrect) existing, functioning code by re-defining dereferencing post facto. I can scarcely see how this marketing-driven syntax change was worth it.

JD

Subject: Re: New Object Method Invocation Syntax Brokenness
Posted by David Fanning on Fri, 20 May 2011 04:07:53 GMT
View Forum Message <> Reply to Message

Folks,

I thought this discussion was clearly important, but I didn't expect to be personally involved in it. But just tonight I received a report from a user about one of my very old programs. What do you know! *Exactly* this problem rearing its head.

I would *never* have been able to figure this out!

Thanks to everyone for sheding light on this. I think now this syntax change might cause more problems than I previously anticipated.

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")
