Subject: Re: Resolving Built-ins and FORWARD_FUNCTION
Posted by JD Smith on Tue, 09 Dec 2003 23:46:27 GMT
View Forum Message <> Reply to Message

On Tue, 09 Dec 2003 14:05:23 -0700, Wayne Landsman wrote:

>> I'm not sure why nobody complained: the bug is present as far back as
>> v5.5 (which is the earliest version I had to test). The test is easy,
>> if you have AstroLib:
>>
>> IDL> .run writefits
>> IDL> resolve_all
>>
>> will give an error.
>>
>>
> I'm sure this has been mentioned but another workaround is to use
>
> IDL> resolve_all,/continue_on_error
>
> which is what I've been using by default. The disadvantage is that
> you might not recognize when a real procedure is missing. --Wayne

Thanks Wayne. I was wondering why FORWARD_FUNCTION was needed at all,
so I took a look. I think I've found the reason: when IDL compiles a
routine, it takes any function call with keyword arguments (which it
*knows* is a function call, and not a variable using the old ()
indexing syntax), and checks that it really exists somewhere as a
function... it doesn't compile the function, it just checks that it
exists. Example:

```
;; Compiles fine, since it could be an indexed variable, for all IDL
;; knows
pro testff
  if !PI eq 2.0 then ret=unknown_function(b)
end

;; Syntax error on compile, since it's not a known function, and
;; indexing statements shouldn't have KEYWORDS in them!
pro testff
  if !PI eq 2.0 then ret=unknown_function(b,TEST=2)
end

;; Compiles fine
pro testff
  FORWARD_FUNCTION unknown_function
  if !PI eq 2.0 then ret=unknown_function(b,TEST=2)
```

```
end

;; Compiles fine
pro testff
  COMPILE_OPT IDL2
  if !PI eq 2.0 then ret=unknown_function(b,TEST=2)
end
```

Notice that it doesn't matter that the function will *never* be called (in Euclidean universes, anyway). Also, amusingly, IDL really only checks that the function in question is built-in or that there is a file named "unknown_function.pro" somewhere on the !PATH: this could contain a procedure named "unknown_function", or your grandmother's bourbon fruitcake recipe, so long as it exists.

If you call a function with keyword arguments which doesn't exists (e.g. because it's available only in a later version of IDL), you can use FORWARD_FUNCTION to keep IDL from issuing a syntax error, otherwise it will insist that, if it's not a function, it must be an indexing statement, with a keyword-like syntax error inside.

Interestingly, if you use "COMPILE_OPT IDL2", this type of error disappears. This is because IDL no longer needs to go over every thing that looks like foo(), and check that it's either a function or a syntax-error-free indexing statment. It just assumes it's a function. This probably speeds up compiling just a bit too, so I'd go as far as to say COMPILE_OPT IDL2 is probably a better all-around solution than FORWARD_FUNCTION, unless you require IDL<v5.3 compatibility.

JD

---