Subject: Re: Resolving Built-ins and FORWARD_FUNCTION
Posted by JD Smith on Tue, 09 Dec 2003 19:29:26 GMT
View Forum Message <> Reply to Message

On Mon, 08 Dec 2003 23:15:20 -0700, Craig Markwardt wrote:

> JD Smith <jdsmith@as.arizona.edu> writes:
>
>
>> Sent to RSI:
>>
>> ==
===============
>> Using FORWARD_FUNCTION creates an unresolved stub in the routine list,
>> even for built-in routines. E.g., in the NasaLib WRITEFITS you find :
> ...
>> IDL could either check for built-in's being used in FORWARD_FUNCTION,
>> or RESOLVE_ROUTINE could do the same, or FORWARD_FUNCTION functions
>> could be removed from the list once they are encountered in the file.
>> Since you can't override a built-in command (like FILE_SEARCH) with any
>> amount of !PATH fiddling, it makes sense not to put built-ins on the
>> unresolved list via FORWARD_FUNCTION.
>> ==
===============
>
> Yes, these seems like a totally legit complaint. I don't think this
> problem showed up before IDL 6.0, did it? Or else, why did nobody
> complain before now?

Thanks Craig. RSI has filed an internal bug-fix request on this one,
and suggested a workaround of using "COMPILE_OPT IDL2" in place of
FORWARD_FUNCTION. Of course, this would not really help Wayne, who is
using FORWARD_FUNCTION to allow NasaLib to run for older IDL 5.x
versions (COMPILE_OPT was introduced in v5.3). And it also doesn't
help if you're "compiling" in code from libraries over which you have
no control.

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.

>> Also, does anyone know what a SAV file run in the IDLVM does with a

>> statement like:
>>
>> source=routine_info('MyPro',/SOURCE)
>>
>> I use these types of constructs to locate data bundled with my source
>> distribution, and I want it to work with the IDLVM too. Since the VM
>> technically doesn't do any compiling of files, I presume it might not
>> do any path searching for file source either, in which case I'd have to
>> come up with something different.
>
> Why not try it yourself? I found that the "source" it reported was the
> path of the .sav file.

Because I knew I could get you to do it for me ;). Next question I
should probably find out for myself: is there a programmatic way to
tell if you're running from a restored SAV file or from real, live
source? Can you tell I've almost never built a routine SAV file? I'm
trying to see if I can get a large package to run with the IDLVM.

Thanks,

JD