Subject: Functions and arrays
Posted by thompson on Tue, 03 Dec 1996 08:00:00 GMT
View Forum Message <> Reply to Message

I've just run into a serious limitation in the way IDL handles functions and arrays. I've always been aware that there was an ambiguity in the IDL syntax, in that function calls and array indexing are the same. However, up till now I had always thought that IDL handled that ambiguity in a consistent manner. Now I realize (or relearn) that it does not.

Consider the following set of routines:

```
 function test1, i
 test2 = [1,3,7]
 a = test2(i)
 return, a
 end
```

and

```
 function test2, a
 return, a-1
 end
```

It turns out that the behavior of the TEST1 routine critically depends on the order in which routines are compiled. If one compiles TEST1 first, then it will use the internal TEST2 variable as it should, even if the TEST2 routine is later compiled. However, if the TEST2 function is compiled before TEST1, then TEST1 will not be able to access its own internal variable.

This "feature" caught us because one of our routines happened to use an internal variable that matched the name of an unrelated function. Sometimes the routine would work correctly, and sometimes it wouldn't. It took quite a bit of detective work to track down the problem.

The most insidious aspect of this behavior is that correctly working software could be brought down by the addition of a new function to the tree, in a completely unrelated area of code.

It is my opinion that IDL should be tightened in its handling of arrays and functions. If a procedure contains a variable with a given name, then it should be unambiguous that one is referencing that variable, and not some function which happens to share the same name. Programmers would need to avoid using a variable name which coincides with a function they wish to reference within the same procedure--which is just good programming practice in any case.

William Thompson

Subject: Re: Functions and arrays
Posted by Peter Mason on Tue, 10 Dec 1996 08:00:00 GMT
View Forum Message <> Reply to Message

On Sat, 7 Dec 1996, David R. Klassen wrote:
> See, right now my problem is just the opposite. I reference a function
> that IDL is trying to interpret as an array even though the "array" has
> not been previously referenced. I can't seem to understand why the
> program can't retrieve the function...

Here are some observations that might help you sort this out:

It appears that IDL always makes the distinction between variables and functions at the time a routine is "compiled". This distinction has to be made if an identifier, say XXX, appears with brackets somewhere in the routine - as in XXX(...). (I must say that I was a little surprised to find this out. I've always considered IDL to be an interpreter with some sort of "tokenising" pass - not a true compiler, and thought that the determination of exactly what XXX is would usually be done at run-time. Anyway...)

IDL appears to follow these rules to work out whether XXX is a variable or a function:
1 If a function XXX has already been compiled, or declared in a FORWARD_FUNCTION statement somewhere, then it remains a function. (If a PROCEDURE XXX has been compiled, it is not noticed here. IDL can live with both a function and a procedure named XXX!)
2 If XXX appears in a statement which could assign a value to XXX, BEFORE the first statement which contains XXX(, then IDL decides that XXX is a variable. (IDL does appear to handle statements like SOME_ROUTINE,...,XXX as potential assignments, along with XXX=.)
3 If IDL first encounters XXX in the form XXX(, then it goes looking for a file XXX.pro (or XXX.sav, I guess). If such a file is found, it doesn't compile it, but it decides that XXX is a function. (IDL is expecting FUNCTION XXX to be the only or last routine defined in the file XXX.pro. It doesn't appear to check that this is actually the case, at this stage.) If it doesn't find such a file, then it decides that XXX is a variable.

The simplest thing for you to do would be to use a FORWARD_FUNCTION statement. (You can use a FORWARD_FUNCTION statement anywhere, as long as IDL sees it before it attempts to compile a routine which refers to the function in question. A handy place is in IDL's startup file.)

Peter Mason