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