Subject: Re: IDL wish list (continued)
Posted by Ken Knighton on Wed, 13 Dec 1995 08:00:00 GMT
View Forum Message <> Reply to Message

rfinch@water.ca.gov (Ralph Finch) wrote:
>
>> This is part of the fundamental nature of IDL and would be
>> difficult if not impossible to resolve without the use of either
>> explicit or implicit typing.
>
> Not really, all you have to do is arrange for the interpreter to
> warn users that a variable they are using has not been explicitly set
> to a value. MATLAB does this and doesn't require typing.

I don't know anything about MATLAB, but IDL already warns the user if he attempts to use an undefined variable:

IDL> print, a
% PRINT: Variable is undefined: A.
% Execution halted at: $MAIN$
IDL>

Iarla's post said:

> How about an IDL that would not create an undefined variable whenever
> something gets mistyped...

Allocating a data name when an IDL line is compiled is part of the basic language system. IDL first compiles the code and then executes it. Part of compilation includes putting entries into the data name table. In order to prevent the variable from being created, the line would have to be compiled, and if the above error occurred when the line was executed, the variable would have to be removed from the name table. IMHO, this would be either difficult or impossible to implement and certainly not cost effective. Also, would it be backwards compatible?

Iarla's post doesn't give a lot of information, but as I see it, the two problems that occur when undefined variables are created are:

1. The user mistypes a variable name interactively at the main program level and receives the above warning. In addition, a new variable is created which counts against the local variable limit and clutters up the HELP listing. This could be helped by allowing more local variables. The user can also use the DELVAR procedure to delete the undefined variables; however, this has the side effect of erasing the current main program.

or

2. The user creates a function or a procedure and mistypes the variable name. The function or procedure compiles correctly, but when it is executed, a run time error similar to the above occurs. The user then has to edit, compile, and run again. This cycle continues until no more run time errors of this type occur.

Number one is of very little concern to me personally, but two is extremely annoying and wastes a lot of my time. The way that most compiled languages get around this problem is to require static typing. This enables these problems to be discovered at compilation time and enables numerous errors to be discovered in one compilation cycle. IDL could be improved as an application development language by:

1. Providing an option to do strong typing. Probably not cost effective.

2. Making the compiler more intelligent so that it can recognize this type of problem and issue a warning.

3. Providing a lint type of tool to check source code for potential problems that should be looked into.

Ken Knighton knighton@gav.gat.com knighton@cts.com
Fusion Division
General Atomics
San Diego, CA
