Subject: Re: Global variables and command line
Posted by J.D. Smith on Fri, 17 Apr 1998 07:00:00 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
>
> Daniel SAGE (daniel.sage@epfl.ch) writes:
>
>> I would like use the variables created in the input command line inside
>> the procedures/functions. But in my application where I automatically
>> generate IDL code, I can't use the common mechanism or passing by
>> parameters.
>
>> How is it possible to make V2 visible inside the test procedure when I
>> can't pass the parameters with the run procedure ?
>
> Well, at the risk of being too obvious here, I would suggest
> putting V2 *in* the common block. :-)
>
>> IDL> common var, v1
>
> to IDL> common var, v1, v2
>
> Cheers,
>
> David
>
> ---
> David Fanning, Ph.D.
> Fanning Software Consulting
> E-Mail: davidf@dfanning.com
> Phone: 970-221-0438
> Coyote's Guide to IDL Programming: http://www.dfanning.com/

At the risk of being too arcane here, I think Daniel's point is quite a valid one, and his problem less easily solved than it might at first glance seem. Sometimes, one wants to access variables from the $MAIN$ level in a routine, and sometimes one wants to put variables created inside a routine into $MAIN$. You might wonder what circumstance could exist that would require this functionality. I would simply then direct you to RSI's own Insight, in which after getting in you have the option of "Select Data to Import"..."IDL Variables"... with a list of $MAIN$ level variables displayed. Now clearly the folks at RSI have convenient access to the names and data locations of the $MAIN$ level variables, and I think they should give us mere ordinary users that access also. As phrased so judiciously in the Reference Guide under the Help procedure:

The OUTPUT keyword is primarily for use in capturing HELP output in order to display it someplace else, such as in a text widget. This keyword is not intended to be used in obtaining programmatic information about the IDL session, and is formatted to be human readable. Research Systems reserves the right to change the format and content of this text at any time, without warning. If you find yourself using OUTPUT for a non-display purpose, you should consider submitting an enhancement request for a function that will provide the information you require in a safe form.

I am henceforth considering submitting an enhancement request for a function that will provide the names and data locations (e.g. through pointers) of $MAIN$ level variables inside routines, and the ability to create $MAIN$ level variables from within routines, in a safe form.

By the by, for creating $MAIN$ level variables, I used to use this mechanism:

Have a common block which contains information on new variables (including their names) created by a program for export into the $MAIN$ level. Have a $MAIN$ level routine as the only interface into the program, and have it call that program. After the program exits, the remainder of the $MAIN$ level routine simply looks in the common block, and creates any exported variables in $MAIN$ (or, e.g just creates new $MAIN$ level pointers to the heap data). This used to work fine. However, with the advent of the non-blocking widget interface, it doesn't work in conjunction with an active command line (the $MAIN$ level program runs through immediately and exits without creating anything).

Of course, you can always hack something, so my present solution for non-blocking applications is to use a little procedure which sets a pointer into a common block, where it can be retrieved from within the program.

E.g. I might say

mark, thisPointer

and have the data *thisPointer available immediately in the program. Indeed, you could take this idea further and have a single $MAIN$ level "import" routine which looks in the relevant common block for any exported pointers, which have their *names* recorded also in the common block, and executes them into $MAIN$ with the same names. But this requires the user to explicitly run the import routine, which not elegant.

As an example, suppose I have some image in a file, call it A. In a fancy widget program, I load A from disk and edit that image. When I exit the program, it would be ideal if A (or a pointer to the heap data which is A) could be created in the $MAIN$ level (as A) if I so choose, for my further analysis and editing. Currently, I would have to run my $MAIN$ level import routine to accomplish this.

Perhaps RSI is trying to protect us from ourselves here, realizing that with the problems people have with variable locality already, breaking it slightly might serve to confuse even more. But they really shouldn't underestimate our ability to harness and control whatever new power comes our way. And besides, if they get to do it, then so should we.

Alright, the rant is over.

JD

--

J.D. Smith |*| WORK: (607) 255-5842
Cornell University Dept. of Astronomy |*| (607) 255-4083
206 Space Sciences Bldg. |*| FAX: (607) 255-5875
Ithaca, NY 14853 |*|