Subject: Re: Arg_Present, XSurface, and Other Assorted Blunders
Posted by davidf on Fri, 08 Aug 1997 07:00:00 GMT
View Forum Message <> Reply to Message

Kirt Schaper writes in response to my longish article:

> I might be missing something here, but it seems to me that 'arg_present'
> is doing precisely what it is supposed to do, report whether the
> variable passed to it is (1) an argument to the current routine AND
> (2) a variable into which a value will be copied when the current
> routine exits. If the user called the routine w/o specifying the a
> value for "data", then there is NO way that any value assigned to a
> variable "data" w/in the routine. (This is assuming that they did not
> pass in an unassigned variable as a keyword parameter.)
>
> 'Arg_present' is intended to allow avoiding computation of output
> variables if the calling routine has no way of retrieving those values.

Yes, I am pretty clear now about its rather limited usefulness.
My point is exactly this: its *name* implies things about it that
are not true. Hence, it is likely to be misused by unwary programmers
like me.

Cheers,

David

David Fanning, Ph.D.
Fanning Software Consulting
Customizable IDL Programming Courses
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com

Subject: Re: Arg_Present, XSurface, and Other Assorted Blunders
Posted by Kirt Schaper on Fri, 08 Aug 1997 07:00:00 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
>
> ... alot of stuff and then ...
>
> I looked at the code near the LoadData call and here is
> what I found:
>
> Catch, error
> IF error NE 0 THEN BEGIN ; Can't find LoadData.

>   data = DIST(41)
>   x = Findgen(41)
>   y = Findgen(41)
>   ENDIF
>
>   IF Arg_Present(data) EQ 0 THEN BEGIN
>   data = LoadData(2)
>   ENDIF
>
> Here is what this piece of code is *meant* to do:
> I want to supply some default data if the user doesn't
> pass data into the program in the argument "data".
> If I need to create some data I want to use my LoadData
> program to get it. But I also know that some people won't
> have LoadData, so I want to "Catch" the error that happens
> when you try to call a program that is not on your path.
> If I catch the error, I create the "data" variable and
> continue program execution.
>
> Because this is a program using IDL 5 specific functionality,
> I also wanted to use the new Arg_Present routine to prove
> that I am an up-to-date and with-it IDL programmer.
> Apparently, I forgot to read the documentation. In any case,
> here is what happens.
>
> When the call is first made, "data" is not present and
> Arg_Present reports this correctly. The LoadData error occurs
> and I bounce up to my error handler. I define "data" and
> continue. But Arg_Present *STILL* reports data as missing
> in action. This is so even when it is defined AND a variable
> that is passed by reference. As a result, my code goes into
> an infinite loop.
>
> So now I am confused about exactly what Arg_Present is suppose
> to do, but I do know this: it is a grievous mistake to treat
> Arg_Present as an function that tells you if an argument
> is present! To fix the problem I swallowed my pride and
> went back to the terribly misnamed N_Elements to solve
> my problem.

I might be missing something here, but it seems to me that 'arg_present' is doing precisely what it is supposed to do, report whether the variable passed to it is (1) an argument to the current routine AND (2) a variable into which a value will be copied when the current routine exits. If the user called the routine w/o specifying the a value for "data", then there is NO way that any value assigned to a variable "data" w/in the routine. (This is assuming that they did not pass in an unassigned variable as a keyword parameter.)

'Arg_present' is intended to allow avoiding computation of output
variables if the calling routine has no way of retrieving those values.

To illustrate, consider the following program/output:

```
<<<<<<<<<<<<<<<<<<< start program >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
pro idl_test, arg1, ARG2=arg2

print,'n_params=',n_params()

if (arg_present(arg1))  then print,'arg1 present'  else print,'arg1
absent'
if (keyword_set(arg1))  then print,'arg1 set'      else print,'arg1 not
set'
print,'n_elements(arg1)=',n_elements(arg1)

if (arg_present(arg2))  then print,'arg2 present'  else print,'arg2
absent'
if (keyword_set(arg2))  then print,'arg2 set'      else print,'arg2 not
set'
print,'n_elements(arg2)=',n_elements(arg2)

end
<<<<<<<<<<<<<<<<<<< end program >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

<<<<<<<<<<<<<<<<<<< start output >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
IDL> idl_test
n_params=       0
arg1 absent
arg1 not set
n_elements(arg1)=           0
arg2 absent
arg2 not set
n_elements(arg2)=           0

IDL> help,xxx
XXX             UNDEFINED = <Undefined>
IDL> idl_test,xxx
n_params=       1
arg1 present
arg1 not set
n_elements(arg1)=           0
arg2 absent
arg2 not set
n_elements(arg2)=           0

IDL> help,xxx,yyy
```

```
XXX             UNDEFINED = <Undefined>
YYY             UNDEFINED = <Undefined>
IDL> idl_test,xxx,arg2=yyy
n_params=       1
arg1 present
arg1 not set
n_elements(arg1)=          0
arg2 present
arg2 not set
n_elements(arg2)=          0
<<<<<<<<<<<<<<<<<<<< end output >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
```

So, 'arg_present' (as opposed to 'n_elements') can detect whether there is any reason to supply a OUTPUT variable. It isn't as useful for detecting whether or not an INPUT variable needs to have a default value supplied for it, and 'n_elements' should be used for this.

Don't just do something, sit there!

Kirt Schaper | Bell: (612) 725-2000 x4791
PET Center (11P) | net: kirt@pet.med.va.gov
VA Medical Center | FAX: (612) 725-2068
MPLS, MN 55417 | URL: http://pet.med.va.gov:8080