Subject: Re: string definition question
Posted by Paul Van Delst[1] on Tue, 14 Jan 2003 21:27:01 GMT
View Forum Message <> Reply to Message

Mark Hadfield wrote:
>
> "Paul van Delst" <paul.vandelst@noaa.gov> wrote in message
> news:3E2432EC.18E46318@noaa.gov...
>> ..
>> I'm a bit anal about argument checking in IDL. After establishing that the
> correct
>> number of arguments has been passed using:
>>
>> n_arguments = 1
>> IF (N_PARAMS() LT n_arguments) THEN $
>> MESSAGE, 'Invalid number of arguments.', $
>> /NONAME, /NOPRINT
>
> I see your actual question has been answered by others, so permit me to take
> another tack. Why do you set the NONAME & NOPRINT keywords?

The very first thing I do in *all* my "serious" IDL procedures is this:

```
CATCH, Error_Status
IF ( Error_Status NE 0 ) THEN BEGIN
  CATCH, /CANCEL
  MESSAGE, !ERROR_STATE.MSG, /CONTINUE
  RETURN
ENDIF
```

and in my functions, this:

```
@error_codes
CATCH, Error_Status
IF ( Error_Status NE 0 ) THEN BEGIN
  CATCH, /CANCEL
  MESSAGE, !ERROR_STATE.MSG, /CONTINUE
  RETURN, FAILURE
ENDIF
```

where in the second example, the values for SUCCESS, INFORMATION, WARNING, and FAILURE are
defined in the include file "error_codes.pro".

The *last* thing I do in procedures is:

```
 CATCH, /CANCEL
END
```

and in functions

```
  CATCH, /CANCEL
  RETURN, SUCCESS
END
```

Now - any error checking I do I use something like:

```
  MESSAGE, 'An error occurred! Oh no!', $
         /NONAME, /NOPRINT
```

All this does is set the !ERROR_STATE.MSG which I then actually print out in my CATCH error handler - all errors tripped using the MESSAGE, 'xxxx', /NONAME, /NOPRINT get sent to the CATCH. I do this so I *always* have only one SUCCESSful exit point and only one FAILed exit point.

> And why check
> the number of parameters? Isn't it better to check each argument to see that
> it's been defined (with N_ELEMENTS) or that it's available for output (with
> ARG_PRESENT) as necessary.

I do both. My simple reasoning is if all the required arguments aren't defined then issue an error stating that.

> The additional N_PARAMS check lets you
> distinguish arguments that have been given an undefined value from those
> that are completely missing; I don't think this is a very interesting
> distinction.

Hmm - maybe, but I prefer to err on the side of verbosity. I would rather the error message state "invalid number of arguments" rather than "argument X is not defined" when what really happened was that argument X wasn't even passed into the routine. When the error occurs I want to know *exactly* what occurred - did I forget to pass the argument or did I forget to define it.

> I ask because I feel that I have never really sorted out error checking in
> IDL. I guess that I lean towards a minimal approach: if a piece of code
> requires that a value be defined then I'll learn soon enough if it's not.

Ahh - therein lies the difference in our attitudes. I'm ridiculously anal about checking stuff and issuing error messages every chance I get. I have code consisting of 10's of lines of code and only 1-3 lines are actually the working, non-error checking parts. Minimalism in coding isn't my strong point. :o\

> (It's the code that silently gives you the wrong answer that you've got to
> look out for.)

oh yeah - you betchya. Every programmers nightmare. But your statement that you'll "learn soon enough" if something is wrong is not always the case. A number of times I've found answers to be enticingly correct - only to find out later (sometimes by someone else...gasp! horror!) they were quite bogus.

paulv

--
Paul van Delst
CIMSS @ NOAA/NCEP/EMC
Ph: (301)763-8000 x7274
Fax:(301)763-8545
