Subject: Re: translating an array name to a string
Posted by John-David T. Smith on Thu, 19 Oct 2000 07:00:00 GMT
View Forum Message <> Reply to Message

Craig Markwardt wrote:
>
> Hi JD--
>
> I agree that the use of ROUTINE_NAMES is has some potential problems
> because it is undocumented. I agree with you also that you need to
> protect your usage of ROUTINE_NAMES with a CATCH handler, since there
> are a lot of ways for things to go wrong.
>
> But I don't agree with your implementation :-).
>
> First, the awkward use of CALL_FUNCTION can be avoided by using the
> FORWARD_FUNCTION declaration. This is always safe, even if the
> function being declared is a built-in one.
>
> Second, your check to see if a variable is undefined is rather
> convoluted. It involves two passes to get it right. I prefer instead
> to use the N_ELEMENTS command to immediately determine whether a
> variable is undefined. Unlike *assigning* an undefined variable,
> which does produce an error, simply taking the N_ELEMENTS of an
> undefined variable will not cause an error.
>
> Finally, users need to be aware that the capability to use
> ROUTINE_NAMES to create new variables at another calling level has
> only come with IDL version 5.3. This is documented at
> http://cow.physics.wisc.edu/~craigm/idl/idl.html under Introspection,
> by the way.
>
> So here is my revised version of your code :-) It's shorter and the
> flow control is primarily linear.
>
> ; *******
> forward_function routine_names
>
> catch, err
> if err NE 0 then begin
> catch, /cancel
> message, 'Assign operation failed'
> endif
>
> ; Protect against an already-defined variable
> if n_elements(routine_names(var_name,fetch=1)) GT 0 then begin
> catch,/cancel
> message,'A variable named '+var_name+' already exists.'

```
>  endif
>
>  ; Still here... we need to export ourself to the main level
>  dummy=routine_names(var_name,myvar,store=1)
>  catch, /cancel
>  ; *******
```

This is definitely nicer looking, and it reminded me of a caveat. If you attempt to fetch a variable which doesn't yet exist, an undefined variable will be created on that level for you.

I think our methods offer equal protection against certain types of failure, but I also think call_function provides additional insurance against RSI deciding specifically to remove our capacity to use routine_names() (which they might do if we keep talking about it so much and people catch on!). It is simple to parse *compiler* statements like forward_function for disallowed names. It is impossible (OK, very, very awkward), to prohibit the use of classified *strings*. This is probably paranoid, but that's why I chose call_function. In any case I will modify my method to include the n_elements() test (which I was stupid not to think of).

JD

--

J.D. Smith | WORK: (607) 255-6263
Cornell Dept. of Astronomy | (607) 255-5842
304 Space Sciences Bldg. | FAX: (607) 255-5875
Ithaca, NY 14853 |