
Subject: Re: translating an array name to a string
Posted by [Craig Markwardt](#) on Thu, 19 Oct 2000 07:00:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

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
; *****
;
```

Craig

"J.D. Smith" <jdsmith@astro.cornell.edu> writes:

```
> By the way, for those of you using routine_names for heavy magic... you
> might consider examining the following extra-cautions snippet to export
> a variable to the $MAIN$ level:
>
>   var_free=0
>   catch, err
>   if err ne 0 then begin
>     ;; An undefvar indicates routine_info ran and
>     ;; the variable is free
>     if !ERROR_STATE.NAME ne 'IDL_M_UNDEFVAR' then begin
>       catch,/cancel
>   message,"Can't complete operation... Try obj=sp_sel()"
>   endif
>   var_free=1
>   endif
>
>   ;; If we need to check if the variable name is available, do so.
>   if var_free eq 0 then $
>     rn=call_function('routine_names',var_name,FETCH=1)
>
>   if n_elements(rn) ne 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
>   rn=call_function('routine_names',var_name,myvar,store=1)
>
> basically the idea is to wrap routine_names as a string in
> call_function, to allow your routine to compile even if RSI yanks or
> renames it (it wouldn't compile if you tried to call it directly).
> You'll get an error, of course, which will be caught. You have to
> discriminate between errors caused by the successful operation of
> routine_names(), and those caused by incorrect arguments, changed
> keywords (IDL_M_KEYWORD_BAD), or other mutations routine_names() has
> undergone (such as vanishing altogether -- IDL_M_UPRO_UNDEF). You
> obviously have to have a backup plan too, to tell your users what to do
```

> in case routine_names() has broken. But it's better than your program
> not running at all though.
>
> 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 |

--

Craig B. Markwardt, Ph.D. EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
