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

Craig Markwardt <craigmnet@cow.physics.wisc.edu> writes:
> 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.

Ooops, this is actually a mistake on my part, at least on versions of
IDL earlier than v5.3.

ROUTINE_NAMES(NAME, FETCH=1) will

* succeed if a variable exists and is defined
* return an undefined value if the variable exists but is undefined
* utterly fail if the variable doesn't exist, stopping execution

This doesn't do what you want. However, I think the better approach
than using CATCH, is to use the VARIABLES keyword with ROUTINE_NAMES
to find out if the variable exists first. This is my revised version.

```
; *******
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
vnames = routine_names(variables=1)
wh = where(strupcase(var_name) EQ vnames, ct)
if ct 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 B. Markwardt, Ph.D. EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
---------- ----------