Subject: Re: Keyword checking
Posted by thompson on Wed, 29 May 2002 14:55:20 GMT
View Forum Message <> Reply to Message

Paul van Delst <paul.vandelst@noaa.gov> writes:

> Randall Skelton wrote:
>>
>> I have run into a great deal of trouble checking keywords in IDL and I
>> thought I would relay my thoughts and frustrations. I am trying to
>> prevent a user from passing an undefined variable in a keyword...
>>

(stuff deleted)

> Why the arg_present? Do you specifically want to tell the user that what they
> passed was undefined? I thought arg_present was for using when you wanted to
> return something in the variable? If the data is for input only, won't
> n_elements() suffice?

> me confused.

I agree. I can't think of any legitimate reason why one would want to generate an error message if an undefined keyword was passed.

In the case of an input keyword, and undefined keyword should be treated exactly as if the keyword was not passed at all. The reason for this is quite simple. If one has embedded subroutines with keyword inheritance, there has to be some way to pass the keywords along. For example

```
pro test1, key1=key1
test2, key1=key1
return
end

pro test2, key1=key1
if n_elements(key1) ne 0 then help, key1 else print, 'KEY1 not passed'
return
end
```

This obviously very trivial example is enough to illustrate my point. If the user calls the procedure TEST1, the keyword KEY1 will be passed along to TEST2 whether or not the user called TEST1 with that keyword. If the user simply calls TEST1 without the keyword, it will still appear in TEST2 as an undefined value. There's nothing wrong with that.

In the output case, all one cares about is whether or not there's a variable to receive the output. It doesn't matter if the variable was defined previously

or not.

I can only conclude that trapping an undefined keyword as an error is bad IDL programming practice.

William Thompson
