Subject: Re: string definition question
Posted by JD Smith on Tue, 14 Jan 2003 19:36:42 GMT
View Forum Message <> Reply to Message

On Tue, 14 Jan 2003 11:45:03 -0700, William Thompson wrote:

> Paul van Delst <paul.vandelst@noaa.gov> writes:
>
>> mwvogel wrote:
>>>
>>> As my news server refuses my post, I'll paste it here :-)
>>>
>>> ////////////////////
>>> I would try KEYWORD_PRESENT; with A defined as 'IDL', B as '' and C
>>> undefined I get the following :
>>> IDL> A = 'IDL' & B = '' & PRINT, KEYWORD_SET(A), KEYWORD_SET(B),
>>> KEYWORD_SET(C)
>>>
>>> 1 0 0
>>>
>>> I guess that works in routines too.
>
>
> I've always been disappointed that the KEYWORD_SET() routine does not
> follow the same logic as the rest of IDL for deciding whether something
> is true or false. According to the definition of true and false in the
> documentation
>
> Definition of True and False
>
> The condition of the IF statement can be any scalar expression. The
> definition of true and false for the different data types is as
> follows:
>
> * Byte, integer, and long: odd integers are true, even integers are
> false.
>
> * Floating-Point, double-precision floating-point, and complex:
> non-zero values are true, zero values are false. The imaginary part of
> complex numbers is ignored.
>
> * String: any string with a nonzero length is true, null strings are
> false.
>
> However, the KEYWORD_SET() documentation simply says
>
> The KEYWORD_SET function returns a nonzero value if Expression is
> defined and nonzero or an array, otherwise zero is returned. This

> function is especially useful in user-written procedures and functions
> that process keywords that are interpreted as being either true
> (keyword is present and nonzero) or false (keyword was not used, or was
> set to zero).
>
> In other words, KEYWORD_SET() treats integer and floating point equally,
> while they're treated differently in conditional statements. I've
> always found that troublesome. On the other hand, the treatment of
> strings is consistent between the two, although it's undocumented for
> KEYWORD_SET().

Strangely enough, this is precisely the reason I *do* like KEYWORD_SET.
Had IDL inherited a more useful definition of TRUE and FALSE than the
FORTRAN versions, a separate logic for KEYWORD_SET wouldn't be necessary,
but do you really want to test for non-zero status in your keywords with:

```
 if keyword_set(key) then if key gt 0 then do_something
```

This would not really be a savings over:

```
 if n_elements(key) gt 0 then if key gt 0 then do_something
```

And the only time you'd profit from the altered definition would be
discriminating even/odd integers... hardly that common an operation
(for me at least):

```
 if keyword_set(key) then print,"It's odd"
```

which in real IDL would need to be:

```
 if n_elements(key) gt 0 then if key then print,"It's odd"
```

I agree that the variety of TRUE/FALSE meanings scattered throughout
IDL is somewhat disconcerting, but in this case, I think it's well
worth it!

JD
