Subject: Re: string definition question
Posted by thompson on Wed, 15 Jan 2003 15:31:53 GMT
View Forum Message <> Reply to Message

JD Smith <jdsmith@as.arizona.edu> writes:

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

(stuff deleted)

>> 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:

(stuff deleted)

But I don't want to test for non-zero status! I want to test for *Boolean*
status--that's what KEYWORD_SET() is supposed to be for! The current
KEYWORD_SET() fails to correctly treat boolean parameters formed out of
operations such as AND, OR, and NOT. Try this in an IDL program

```
A = 3
B = 3
TEST_EQUAL = A EQ B
MYPROC, MYKEYWORD=(NOT TEST_EQUAL)
```

and see what you get for KEYWORD_SET(MYKEYWORD). I know I've been bitten by
that one.

> 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!

And I agree that the definition of TRUE/FALSE used in IDL's Boolean logic is
somewhat byzantine, but the problem is created by using different definitions
in different places.

Perhaps KEYWORD_SET() should have a /BOOLEAN keyword to force compliance with
how True and False are used elsewhere in IDL.