Subject: Re: Order of argument evaluation (Was: Re: making a checkerboard array?)
Posted by Allan Whiteford on Wed, 26 Sep 2007 16:51:06 GMT
View Forum Message <> Reply to Message

Fï¿½LDY Lajos wrote:
>

<snip>

>
> In C/C++/Fortran/... there is a rule that a memory cell can be changed
> at most once in a statement, otherwise the result is undefined (and so
> different implementations can create different results without violating
> the standard). I guess a similar rule is in effect in IDL, too.
>

That's part of the question... does such a rule exist for IDL? You can change the same variable twice in Perl (the implementation is the standard etc. etc.) even with strict and warnings on it doesn't complain about something like:

```
$c=3;
$d[$c++]=$c++;
```

I've never found anything in the IDL manual which explictly says either way what is allowed in IDL. Maybe I should write to the nice support people and ask if they have an opinion.

> These examples give syntax error prior IDL 6.0:
>
> IDL> print, !version
> { sparc sunos unix 5.3 Nov 11 1999}
> IDL> ((a=fltarr(3)))[1]=10 & print, a
> % Expression must be named variable in this context: <FLOAT Array[3]>.
> % Execution halted at: $MAIN$
> IDL>
>

In IDL 6 they changed it so that setting a variable returns the variable set rather than the value of the variable set (i.e. an expression):

```
IDL6> help,(a=fltarr(3))
A               FLOAT     = Array[3]

IDL5> help,(a=fltarr(3))
<Expression>    FLOAT     = Array[3]
```

I wonder if this means that they think it's ok for things like ((a=fltarr(3)))[1]=10 to be written or if they changed it for some other purpose. Again, maybe a question for the support people.

Thanks,

Allan

> regards,
> lajos