Subject: Grumbling about setting double precision
Posted by wlandsman on Tue, 20 Dec 2011 16:58:34 GMT
View Forum Message <> Reply to Message

I am converting a Python program to IDL which includes several hundred double precision data values. In Python the data values are written as follows:

a = [2721.64363164731, -1615.73035059635]

As described by David Fanning in http://www.idlcoyote.com/math_tips/double.html to make these double precision in IDL, it is not enough to convert the vector to double

```
IDL> print,double(a),f='(2f18.10)'
  2721.6435546875  -1615.7303466797
```

and it is not enough to put a "d" at the end of one of the values

```
IDL> a = [2721.64363164731d, -1615.73035059635]
IDL> print,double(a),f='(2f18.10)'
  2721.6436316473  -1615.7303466797
```

Instead, I must spend an hour in my editor putting a "d" at the end of every single data value.

But it would sure be nice to be able to somehow tell the IDL compiler to interpret numeric values as double precision. --Wayne

Subject: Re: Grumbling about setting double precision
Posted by Kenneth P. Bowman on Wed, 21 Dec 2011 18:44:55 GMT
View Forum Message <> Reply to Message

In article <MPG.295bcfa9c9a365f0989962@news.giganews.com>,
David Fanning <news@dfanning.com> wrote:

> Kenneth P. Bowman writes:
>
>> David, you're always such a pessimist!
>
> I'm just saying, my primary motivation is to prevent
> repetitive strain disorder. ;-)
>
> Cheers,
>
> David

And, to be clear, my solution makes sense if you have a block of contiguous constant definitions, or a small number of source programs with constants located at the top of the program, but that assumes reasonably well written code.

I guess I would be pessimistic too.

Ken

Subject: Re: Grumbling about setting double precision
Posted by Karl[1] on Thu, 22 Dec 2011 20:08:23 GMT
View Forum Message <> Reply to Message

Editing: A lot of times, constant arrays like this come from someplace else and someone wrote a script or program to generate them. Such a script or program can be modified to add the 'd'. If that is not feasible, there are plenty of editors that support keyboard macros which can be run repeatedly for a given number of times. Still tedious but not as bad.

Regular expressions are also possible. In Notepad++:

Find: ([0-9]*[\.]*[0-9]*)([,\]])
Replace: \1d\2

Press Replace All and you're done. If that make you nervous, you can restrict the range of Replace All by selecting the text and checking the "In selection" box.

Memory: It is probably a mistake to default to double. It would take up more space and cause other problems. Variables that are created as a result of this array being part of the calculation would be promoted unnoticed to double as well, increasing memory usage even more. And the change in precision may affect results.

I do like the

b=[24.5d,9999.9]

approach. It expresses intent and doesn't seem that confusing (to me). It isn't perfectly backwards-compatible, but if someone wrote the above, they probably wanted a double array anyway.

It should be easy to do in IDL. IDL needs to analyze the first array element in order to decide what sort of array to make; int, string, or float. It would surely know if the literal is double or float at that point.
