Subject: Randomu() behavior - BAD!
Posted by Conor on Fri, 13 Nov 2009 19:01:07 GMT
View Forum Message <> Reply to Message

So I did a simple test and have a better idea of how the seed initialization works with randomu(). In my mind, this is not good news! It seems that IDL does have some sort of hidden global "seed" variable. The first time you pass an undeclared variable to randomu and let it initialize the seed, it is storing the seed it generated in this hidden variable, and using that same hidden variable the next time randomu() is called with an undeclared variable. Why is this a problem? Because if you are calling randomu() with both a declared seed variable and undeclared seed variables, it will end up returning the same set of random numbers. This will happen of course if you have one routine that is generating random variables, and you are calling subroutines that also generate random variables. If you don't pass your seed value to the subroutine, then it will be working with an undeclared variable and will end up re-generating the exact same random numbers you've already used!!!! Here's an example:

```
print,randomu(seed,1)
print,randomu(first_var_undeclared,1)
print,randomu(seed,1)
print,randomu(second_var_undeclared,1)
print,randomu(seed,1)
print,randomu(third_var_undeclared,1)
print,randomu(seed,1)
```

Outputs:

```
    0.456874
    0.528637
    0.528637
    0.783314
    0.783314
   0.0769241
   0.0769241
```

Notice that there is no *visible* relation between my calls to randomu () with the seed variable and my calls to randomu with a series of undeclared variables. However since my seed variable was seeded by the same hidden variable used in all calls to randomu(), I end up getting duplicate results from subsequent calls to randomu() - not good! Moreover this hidden variable is global and is used in calls in subroutines as well. Try out these routines:

```
pro test2,seed=seed
 print,randomu(seed,1)
```

```
end

pro test

test2,seed=seed
print,randomu(seed,1)
test2
print,randomu(seed,1)
test2
print,randomu(seed,1)
test2

end
```

When I run 'test' I get the following output:

```
    0.0108795
     0.292802
     0.292802
     0.121008
     0.121008
```

Just to really prove the point, I can then duplicate these numbers by calling randomu() from the main command line using the same seed variable left over from my previous example:

```
print,randomu(seed,1)
print,randomu(seed,1)
print,randomu(seed,1)

    0.0108795
     0.292802
     0.121008
```

I really wish I had understood this behavior before (it is *NOT* documented). I have had routines which keep track of their own seed and call subroutines which use an undeclared seed. That means that my main routine and my subroutine are re-using the same random numbers, which is decidedly bad. Ultimately this is my fault - I should have been passing the seed back and forth between subroutines. However, it would have been nice to have this documented and to have a nice warning about this behavior, because naively I don't expect different programs and functions to be able to communicate expect as I enable them to communicate. Basically RSI has created a hidden global variable and not told anybody about it - that seems like a big lapse on their parts.

For the lazy programmer you could turn this around and make it a good

thing. I'm sure it would be horrible programming practice, but if you just always passed an undeclared variable to randomu() as a seed, it would automatically use this hidden global variable and you wouldn't have to worry about passing your seed back and forth :) That would save you some trouble until the next version update when they change the behavior again without letting anyone know :(

Also, this still doesn't explain the problem I was having in the other thread.
