Subject: Re: Random number generators in IDL
Posted by steinhh on Thu, 06 Feb 1997 08:00:00 GMT
View Forum Message <> Reply to Message

In article <davidf-ya023080005029712392300001@news.frii.com>, davidf@dfanning.com (David Fanning) writes:

[..]
|> IDL random number generators, like most (all?) random number
|> generators on computers, produces a pseudo-random sequence
|> of numbers. If you always start with the same seed, you will
|> always get the same pseudo-random sequence.

I've been told that in some (extreme) situations (like
simulations of radio astronomy observations etc) what they have
to do (well, had to do some years ago, anyway) is to
"tune in" an amplifier to the thermal noise in a resistor,
to be sure they have no systematic effects. I believe military
spec's for testing things with noise also refers to similar
methods - i.e., "true" random generators.

|>
|> Widget programs sometimes run into random number problems
|> if the event handler depends on a random number sequence.
|> If you go into the event handler quickly, over and over, you can
|> appear to get the same number out of the generator. The trick
|> here is to store the seed in a common block (yuck!).

Actually, though I share your dislike of common blocks, this
is one subclass (of a "cache" type common block) of a situation
where I could tolerate common blocks - as long as the whole thing
is well kept away in a separate random function, and with a common
block name that clearly reflects that fact (to avoid clashes).

It's really a question of whether you look at the seed
argument as a nice feature (you can repeat a sequence tripping
an error by using the same seed over again), or as a problem
(how to ensure that it's not the same number over and over again).

|> Or, if you
|> hate common blocks as much as I do, store the seed in the
|> info structure. (This way you can have many independent
|> random number processes going all at once!)

You could have many independent processes doing *exactly*
the same, though ;-)

Keeping the random number in the info structure does allow

consistent "replays" of pseudorandom events without having
all independent processes going, though. And to end up with
the same seed number in many widget applications, you'd need
a fast machine and a "slow" clock.

The only way I know of to guarantee (well, almost) that two seeds
are not the same in two subsequent calls to randomu/randomn
(without storing/comparing them) must be to wait so and so many
seconds to guarantee that the system time used for initialization
is different enough.

Stein Vidar