Subject: Re: Naive pointer question ?
Posted by Craig Markwardt on Tue, 29 Jan 2002 14:28:01 GMT
View Forum Message <> Reply to Message

the_cacc@hotmail.com (trouble) writes:

> Craig Markwardt <craigmnet@cow.physics.wisc.edu> wrote:
>> Yes, that can be a good way too, especially when there can be multiple
>> "sessions."
>>
>
> I'm not sure what people mean when they talk about multiple sessions
> and COMMON block memory being corrupted/overwritten. Countless times
> I've had several xterms open running the same IDL code (containing
> COMMON blocks) on the same machine and never had a problem.
>
> Obviously, multi-threading within a single IDL session will be
> vulnerable but this was only made available in the latest release so
> they can't mean this. Can they ?

Hey "trouble" --

The basic idea is this. Common blocks are a great way to store globally accessible information in one place. The disadvantage is, they store global information in one place.

Let's say you have a widget application that stores its information in a common block. Maybe you're not familiar with these types of applications in IDL, but basically you need some kind of memory area, which keeps information about widget ids, values selected by the user, etc.

Now if you choose to keep that information in a common block, then you've basicaly guaranteed that you can never have more than one widget of that type on the screen at the same time. The reason? A common block can only store one set of information at a time (unless you get really clever). One widget will overwrite the other's data.

Another example is the (currently obsolete) routine called WRITE_GIF, which allows you to write multiple gifs images into a single file. The way this is accomplished is with a little common block inside WRITE_GIF, which keeps the file unit information. There is a problem here, namely you can't have more than one gif file open at a time.

My point was to *use* the fact that a common block can store global and/or persistent information, but *recognize* that it is inherently vulnerable to modification from the outside.

* maintain tight access control over common block
  (give common a unique name; make only one point of access to user)
* enforce programming practices
  (allow as few procedures as possible to write to common block;
   consider that common may be modified when it leaves your control)
* use sparingly
  (use other data structures more appropriate if possible)

The use sparingly part is, if you can find something other than a common block to accomplish the same thing, then use that instead. In the case of widgets, that would be the use of a widget structure stored in the UVALUE, that was pioneered by our very own David Fanning.

Craig

--

---------- ----------
Craig B. Markwardt, Ph.D. EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
---------- ----------