Subject: Re: Dereferencing a large array in a structure
Posted by Braedley on Wed, 27 Sep 2006 20:08:37 GMT
View Forum Message <> Reply to Message

JD Smith wrote:
> On Wed, 27 Sep 2006 05:43:06 -0700, greg michael wrote:
>
>
>> In my widget programs, I settled for the option of putting everything
>> into a single state variable, s - a sometimes huge structure containing
>> all kinds of stuff, and frequently pointers to variable-sized or large
>> items, which gets passed around via the tlb uvalue (extract and set it
>> as the first and last things in the event handler). Then nearly every
>> routine passes s as its first parameter. Seems to me much cleaner than
>> using common blocks. In all other widget uvalues, I pass only its name
>> (e.g. 'tab3.some_button'), and where necessary decompose that to call
>> the right routine.
>
> You've basically re-implemented parts of the object framework which exist
> in IDL already. In that case, a special variable "self" gets passed to
> all class methods magically, by reference. This is IMO by far the nicest
> way to do GUI programming.
>
> The only trick is getting XManager or WIDGET_EVENT to pass events to the
> object methods (normally they only like plain routines). The trick is
> simple and has been well documented: just save "self", the object
> reference, in the TLB UVALUE, and use a tiny glue procedure like this as
> the EVENT_PRO:
>
```
> pro myclass_event,ev
>   widget_control,ev.top,GET_UVALUE=self
>   self->Event,ev
> end
```
>
> On a related note, shouldn't XManager and the event processing in IDL be
> modified to remove this hack? Shouldn't we be able to tell XManager "Pass
> events from this widget heirarchy to method foo of object boo", with bonus
> points for allowing the object/method to change during runtime (similar to
> how widget_control,b,EVENT_PRO= can already be used to change the callback
> routine). This is a simple modification. A better method might be a new
> XMANAGER class, from which objects can inherit to gain all manner of
> (extensible) event processing functionality.
>
> JD

While I agree that these suggestions are elegant and would be great if
I were writing a new program from scratch, I should point out that a)
this is essentially legacy code that I'm adding functionality to and

generally maintaining and b) I don't have time to implement drastic changes (like removing the common blocks) or learn all the nuances of OOP in IDL. The program will need to be overhauled in the future no doubt, and at that time, OOP may be implemented, but now is not that time.

Braedley