Subject: Dereferencing a large array in a structure
Posted by Braedley on Tue, 26 Sep 2006 19:41:54 GMT
View Forum Message <> Reply to Message

In one of my widget programs, I have a tab widget, and the bases that belong to it store a fair amout of data concerning that tab in their uvalues. When a tab is selected, all the data in the uvalue must be dereferenced and loaded into common block variables. This was done so that other widgets within the program have quick and easy access to the data of the currently open tab.

Enter the problem: Some of the data can be very large. One field in the uvalue structure can be as large as a 20 by 50000 double array (or larger), and it obviously can take some time to copy, especially with memory running low with other similarly large structures for other tabs. I've already set the no_copy keyword in calls to widget_control when setting and getting the uvalues to help reduce the load on memory and help out with some speed, but the million element array still needs referencing into the structure to be set, and dereferenced from the structure of the newly selected tab, which is taking up a large portion of the time spent. Are the any faster ways of doing this? Bonus points for reducing the load on memory and not making me rewrite every subwidget (ie not messing with the common blocks).

Subject: Re: Dereferencing a large array in a structure
Posted by greg michael on Wed, 27 Sep 2006 12:43:06 GMT
View Forum Message <> Reply to Message

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.

I think this gets the points for the speed and memory items. But you'd have to mess with your common blocks... (although it might not be so hard to pull out all the variables and put them in a state variable)

Greg

Braedley wrote:
> In one of my widget programs, I have a tab widget, and the bases that

> belong to it store a fair amout of data concerning that tab in their
> uvalues. When a tab is selected, all the data in the uvalue must be
> dereferenced and loaded into common block variables. This was done so
> that other widgets within the program have quick and easy access to the
> data of the currently open tab.
>
> Enter the problem: Some of the data can be very large. One field in
> the uvalue structure can be as large as a 20 by 50000 double array (or
> larger), and it obviously can take some time to copy, especially with
> memory running low with other similarly large structures for other
> tabs. I've already set the no_copy keyword in calls to widget_control
> when setting and getting the uvalues to help reduce the load on memory
> and help out with some speed, but the million element array still needs
> referencing into the structure to be set, and dereferenced from the
> structure of the newly selected tab, which is taking up a large portion
> of the time spent. Are the any faster ways of doing this? Bonus
> points for reducing the load on memory and not making me rewrite every
> subwidget (ie not messing with the common blocks).

Subject: Re: Dereferencing a large array in a structure
Posted by JD Smith on Wed, 27 Sep 2006 18:17:18 GMT
View Forum Message <> Reply to Message

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

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

Subject: Re: Dereferencing a large array in a structure
Posted by David Fanning on Wed, 27 Sep 2006 21:35:28 GMT
View Forum Message <> Reply to Message

Braedley writes:

> 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.

Perhaps, but my guess is that you are closer to this time than
you currently think. Especially when you get through overhauling
the darn thing. :-)

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Dereferencing a large array in a structure
Posted by greg michael on Thu, 28 Sep 2006 12:57:34 GMT
View Forum Message <> Reply to Message

Hi JD,

That looks very nice. I get the idea, but not quite the detail. Could
you point me to some of this documentation, or a small example?

many thanks,
Greg