Subject: Re: Memory fragmentation, passing and common blocks
Posted by mberkley on Sun, 23 May 1993 16:50:59 GMT
View Forum Message <> Reply to Message

> On Sat, 22 May 93 17:35:14 GMT, mayor@vaxine.larc.nasa.gov said:
Lines: 59

SDM> 2) I've written quite a large IDL application and have stuck with
SDM> the convention of passing variables among all the modules. With the
SDM> exception of a few very small common blocks for a few widget event
SDM> handlers, I've avoided common blocks. Now after all this development,
SDM> I'm realizing that I'm passing huge arrays back and forth and wondering
SDM> if it would have been better to put these in common blocks. So this is
SDM> a two part question:

There is a third alternative aside from common blocks or passing
everything from routine to routine, and that is to store structures in
the widgets as UVALUES. In the code you pass widget ids (as
required) from routine to routine, and each routine can choose to get
information from the data structure by extracting the structure from
the widget.

Conceptually, the widget id serves as a pointer to the data
structure, which you access by dereferencing.

Advantages:

1. Avoids common blocks, so that your widgets are separated. If your
   widgets share data in a simple common block, then you can only have
   one instance of each widget. (Yes, I know that you can program
   common blocks to avoid this, but it's a pain. I'm talking about
   simple, naive common blocks).

2. Avoids passing large arrays and structures around. Only two copies
   need exist, one in the widget and one for working. Of course, the
   act of extracting the structure from the widget is copying.

Disadvantages:

1. Lots more work.

2. You still end up with at least two copies of large data structures,
   one in the widget and one working copy. By using multiple widgets
   for storage (eg. one widget for state information, two or three for
   large data objects), you can try to limit access to large data

structures to those routines which really need to access that data.

In the case of very large arrays, I usually resort to common
blocks. If I really want to have two or three instances of a
widget with very large data objects, then I use a fixed array of
these objects in a common block and pass around array indices to
each widget as required.

The real solution to our problem would be real pointers in IDL, but
they're not available. Maybe a Christmas present from RSI? :->

Mike Berkley
University of Victoria
mberkley@sirius.UVic.CA

Subject: Re: Memory fragmentation, passing and common blocks
Posted by webb on Sun, 23 May 1993 22:31:36 GMT

View Forum Message <> Reply to Message

mayor@vaxine.larc.nasa.gov writes:
> 2) I've written quite a large IDL application and have stuck with
> the convention of passing variables among all the modules. With the
> exception of a few very small common blocks for a few widget event
> handlers, I've avoided common blocks. Now after all this development,
> I'm realizing that I'm passing huge arrays back and forth and wondering
> if it would have been better to put these in common blocks. So this is
> a two part question:
>
> a) Can passing variables cause memory fragmentation?
>
> b) What exactly do I have to gain or loose if I start putting variables
> in common blocks instead of passing them?

Passing variables does *not* cause memory problems, because IDL passes
variables by reference. In the user's guide, the description of parameter
passing implies that copies are made of actual parameters, then they are
processed, then they are copied back (see section 10-5). However, this does
not really seem to be true. I've tested this by filling up memory with
large arrays till IDL cannot allocate any more space, then attempting to
call a procedure that does not itself need to allocate data -- IDL
successfully calls the routine, implying parameter passing does not cause
data to be copied.

e.g.

declare the routine mtest.pro

```
pro mtest, arg1
print, arg1(0)
end
```

then (on a Sun with 64M of virtual memory)

```
IDL> a = fltarr(2000,2000,/nozero)
IDL> b = a
IDL> c = a
IDL> d = a
% Unable to allocate memory: to make array.
  Not enough memory
% Execution halted at $MAIN$  .
IDL> mtest, a
% Compiled module: MTEST.
  0.000668437
```

This section of the User's guide could use some rewriting!

Peter