Subject: Re: Are pointers faster in structs ?
Posted by davidf on Wed, 25 Aug 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Richard Tyc (richt@sbrc.umanitoba.ca) writes:

```
> I use a large structure for holding all data in the main widget's user
> value.
>
> eg
>  sState = { btndown: 0b, $
>              cannEndSel: 0b,$
>              DataDim: DataDim, $
>              cube: TEMPORARY(cube) }
>
> cube could be large , say bytarr(512,512,20) Now when the user wants to
> load a  differnet data set, I guess I have the choice of doing
> something like:
>
> sState.cube = 0  ; Free memory to cube ?
>
> newcube = Getnewdata()
>
> sState.cube = TEMPORARY(newcube)
>
> Is this any faster/slower or better/worse than using pointers in the
> structure ?
```

It's not really a question of faster/slower or
better/worse. It's not about efficiency so much
as it is about flexibility.

In this case, newcube is certainly the same size
as oldcube. It *has* to be because you are using
structures. But what about if your data collection
methods change 3 months from now, and then newcube
is twice the size of oldcube? This code is history.

That wouldn't be the case if state.cube were a
pointer to the data, instead of the data itself.
So, by using a pointer, you've created code that is
inherently easier to extend and maintain.

> What if I need to initialize the elements of the structure before I have
> the size of data ?

No-can-do with structures. Another advantage of pointers.

In fact, this is the reason they were created, it seems
to me. Otherwise, it would be impossible to create objects,
which are implemented in IDL as named structures. Without
pointers it would be virtually impossible to initialize
the object structure.

> I need to define sState first without knowing the size of cube. Could I
> do something like :

Absolutely!

```
>  cubePtr = PTR_NEW(/ALLOCATE_HEAP)
>   sState = { btndown: 0b, $
>               cannEndSel: 0b,$
>               DataDim: DataDim, $
>               cubePtr: cubePtr }
>
>  Then later in code, knowing cube :
>
>  cube = bytarr(512,512,20)
>  sState.cubePtr = PTR_NEW(cube, /NO_COPY)
>
>  Is this syntax correct ?
```

Well, it's *almost* correct. You would have only
a *tiny* bit of memory leakage from this syntax. :-)

Better, since you already allocated memory on the heap by
using the ALLOCATE_HEAP keyword (making this a pointer
to an undefined variable, rather than a NULL pointer, which
is what I might have done) is this:

```
   *sState.cubePtr = cube
```

If the NO_COPY thing is important (probably is), I would
initialize and fill like this:

```
    sState = { btndown: 0b, $
               cannEndSel: 0b,$
               DataDim: DataDim, $
               cubePtr: Ptr_New() }

   IF Ptr_Valid(sState.cubePtr) EQ 0 THEN $
      sState.cubePtr = Ptr_New(cube, /No_Copy)
```

> Would this be moe efficient then storing cube
> directly in sState ?

I think the efficiencies are about the same. Although
now you don't have to worry much about NO_COPYs when
you are moving your state structure around, since there
is nothing of consequence in there anymore. Just a bunch
of long integers. :-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155