Subject: Re: Surprising Odds and Ends
Posted by davidf on Mon, 14 Aug 2000 07:00:00 GMT
View Forum Message <> Reply to Message

David Fanning (davidf@dfanning.com) writes:

> 3. The HEAP_GC command (which, heaven forbid, you *really*
> shouldn't be using anyway) is dependent on program
> level. For example, in Cleanup routines I want to
> destroy pointers in my info structure. But if the
> program crashes in an event handler, the info structure
> is undefined in the Cleanup routine. In such a case
> I might want to clean up the pointers by issuing a HEAP_GC
> command. But I could never get this to work. Today I
> found out why. The heap apparently exists *only*
> at the main IDL level. If you try to call Heap_GC from
> some other level (e.g., inside a Cleanup routine) the
> command appears to work, but nothing really happens. This
> command can *only* be used from the IDL command line.

Oh, dear. While my explanation is consistent with
the facts, it is too fanciful by far. (I always
did prefer William Blake's burning tiger's eyes
to Occum's Razor.) In fact, HEAP_GC can be called
from any program level. And it really does search
every variable for a heap reference.

Here is what was happening to make me think differently.
(If you really care--and I told you not to be using
HEAP_GC anyway!--I have a test program you can run.)
I crashed my widget program in an event handler with
the info structure checked out with NO_COPY. I stop,
as I am suppose to, in the event handler module where
the info structure *is* defined, which has the pointer
reference inside it.

I now destroy the widget. My cleanup routine gets called
and the code is executed, including the HEAP_GC which
is in there if the info structure is undefined inside
the cleanup routine, which it is. My pointer does not
get cleaned up. The question is, why not?

The reason appears to be that the Cleanup routine is
called from *within* the stopped event handler module.
But when the Cleanup routine exits, I am back in the
event handler module. Of course, the info structure
exists here, so the previous HEAP_GC didn't clean
up the pointer.

If I type HEAP_GC at the IDL command line, the pointer
is still there, because I am *still* in the event
handler. If I type RETALL & HEAP_GC the pointer
does get cleaned up because now I have exited the
event handler module. The RETALL is what made me
think HEAP_GC was program level dependent.

I makes sense to me, sort of, although how and when modules
are getting called has always been the big MYSTERY to me,
and is one of the reasons I think of widget programming
as magical.

The bottom line is that HEAP_GC doesn't belong in
your code (which is what I have been saying ever
since it first appeared on the scene). What you should
do is CATCH errors, and handle things in such a way that
you never have a need for HEAP_GC. And that is what
this chapter in my book is going to be about. :-)

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