Subject: Re: Hard crash
Posted by Jim Pendleton on Sat, 31 Jan 2015 01:24:33 GMT
View Forum Message <> Reply to Message

On Friday, January 30, 2015 at 4:39:51 PM UTC-7, Heinz Stege wrote:
> On Fri, 30 Jan 2015 13:25:45 -0800 (PST), fawltylanguage@gmail.com
> wrote:
>
>> I can confirm that it was there two hours ago. Now it is here:
>>
>> http://www.exelisvis.com/Company/PressRoom/Blogs/TabId/836/ArtMID/2928/ArticleID/14277/Problems-Assigning-LIST-HASH-etc- to-Class-or-Structure-Tags.aspx
>>
>> Read quickly before it changes its address again :-)
>>
> Thank you Lajos. The second link is working for me to. :-)
>
> However I wonder, if the most important point isn't that in the
> comment: "BUT you should use the SKIP keyword to RESOLVE_ALL before
> creating your SAVE file to ensure the "hash.sav" file is NOT included
> in your build."
>
> I don't use lists and hashes frequently, but does it really make a
> change to use OBJ_NEW() instead of LIST() in the class definition?
> Don't misunderstand me, it is always a good idea to be clearly in
> coding. So it may be more readable to use OBJ_NEW().
>
> But what is the difference? When I define two structures
> s1={a:list()}
> s2={a:obj_new()}
> IDL's help command
> help,s1,s2,/str
> prints:
> ** Structure <1606fea8>, 1 tags, length=4, data length=4, refs=1:
> A OBJREF <ObjHeapVar1(LIST)>
> ** Structure <11587c78>, 1 tags, length=4, data length=4, refs=1:
> A OBJREF <NullObject>
> There are object references in both structures s1 and s2. The first
> one s1 is already allocated in the heap, the second one s2 is a null
> object. But somewhere in the code I have to make it a list:
> s2.a=list()
> And then the help command says, that s2 is equal s1:
> ** Structure <11587c78>, 1 tags, length=4, data length=4, refs=1:
> A OBJREF <ObjHeapVar2(LIST)>
> Is there something wrong in my understanding?
>
> Cheers, Heinz

Sorry about the jumping links, folks.

I have no control over that side of things. It may be that if a comment is added after publication, the original URL changes. (I noticed someone recently purchased the idldatapoint dot com domain and that is now pushed to the top of the Google search hierarchy, so watch out if you access IDL Data Point via Google. Who knows what NSFW junk could end up there.)

With respect to your question, Heinz, the problem isn't with respect to what happens at run time in your example. If you never create a SAVE file from your compiled code, and you never redistribute your compiled code to other users of IDL running different versions of the product, you won't run into this problem.

It may still be relevant if you receive a SAVE file from a colleague and your IDL session starts crashing. This could happen if the SAVE file was built on an older version of IDL.

As we've seen, this is a possibility that has been reported in the IDL user community.

It's a feral catastrophe, and not simply a gedankenexperiment.

The key to understanding the problem is that the IDL compiler *must* restore the hash.sav at compile time if a structure tag is defined as LIST(), HASH(), etc. At that point, all is lost* with respect to IDL "forgetting" about the compiled definitions.

In contrast, a variable assignment call in an executable statement can be deferred until run time, such as "self.l = list()", which is where the COMPILE_OPT IDL2 comes in. In combination with RESOLVE_ALL, SKIP=['LIST','HASH', etc.], the contents of hash.sav will be prevented from being included in your own SAVE file, which should be your goal.

Jim P.**

*When I say "all is lost", it's not really. If you're a glutton for punishment you can, in fact, "edit" your SAVE file contents via the IDL_SaveFile class. This is an exercise left for the reader.

**I still work for Exelis.
