Subject: Re: Releasing temporary variables created with IDL_MakeTempArray()
Posted by dg86 on Thu, 16 Apr 2015 16:02:39 GMT
View Forum Message <> Reply to Message

On Monday, April 6, 2015 at 10:52:52 AM UTC-4, David Grier wrote:
> On Friday, April 3, 2015 at 5:08:14 PM UTC-4, Jim P wrote:
>> One other thing to look for... IDL_MakeTempArray() obviously uses some internal bookkeeping above and beyond the underlying malloc() calls to flag data for subsequent heap-freeing operations.
>>
>> Data allocated this way shows up in HELP, /MEM and the MEMORY() function at the IDL level. You can watch the high water mark rise and fall over time if these data are being freed correctly.
>>
>> Data allocated outside the context of IDL or with lower level routines such as malloc() will not be accounted for in HELP, /MEM for example.
>>
>> Depending on your OS, you might use top, Task Manager or Process Explorer to watch the memory use for the entire IDL process. If you find that your process memory grows but HELP, /MEM doesn't show an increase in the high water mark, then the culprit is likely elsewhere in the DLM.
>>
>> Jim P.
>> "I work for Exelis, too"
>
> Dear Jim and Chris,
>
> Thanks for these very helpful pointers. I've been AFK, and will be testing possible solutions
> later this week. I'll follow up with updates on success (or failure).
>
> All the best,
>
> David

I've figured out what works, and what doesn't, but not why. Everything hinges on what I do with the data returned from my DLM.

1. What doesn't work: creating a pointer to the temporary variable that was allocated in C:

```
ptr_free, self._data
self._data = ptr_new(idlpgr_RetrieveBuffer(self.context, self.image), /no_copy)
```

This is a small snippet from the definition of the dgghwpointgrey object. The whole thing is on github at https://github.com/davidgrier/idlpgr . The idlpgr_RetrieveBuffer routine allocates a temporary variable to return an array of image data. The quoted stanza frees the pointer
to the previous image and creates a new pointer to the new image.

After about 20 minutes of acquiring images at 30 frames per second, this stanza causes my system
to lock up hard. No mouse, no keyboard, no network, nothing. Cycling the power is the only way to recover. Strangely enough, HELP,/MEM shows nothing untoward, right up to the point of failure.
Memory usage is constant, and well below the max.

2. What does work: copying the data from the temporary variable that was allocated in C:

    *self._data = idlpgr_RetrieveBuffer(self.context, self.image)

This runs for at least 24 hours without incident. Here, self._data is a pointer to a previously allocated array of the correct size and type to accept the data from the DLM's subroutine.

I'm satisfied enough with this solution, even though I'd've liked to have avoided copying the data for the sake of efficiency.

My one remaining question is: Why does approach #1 fail, and why does it take so long to fail?

All the best,

David
