Subject: Re: Releasing temporary variables created with IDL_MakeTempArray()
Posted by dg86 on Thu, 02 Apr 2015 00:38:54 GMT
View Forum Message <> Reply to Message

On Wednesday, April 1, 2015 at 7:36:36 AM UTC-4, David Grier wrote:
> Dear Folks,
>
> A DLM I've written in C uses
> IDL_MakeTempArray()
> to allocate an array, let's call it ARR, that gets passed back to IDL.
> I'd like to release that temporary variable in IDL without having
> to call a routine in the DLM that invokes IDL_Deltmp() .
>
> 1. Can I just do something like
> IDL> ARR = 0
> to release the temporary variable? Or is there another IDL-based method?
>
> 2. At a deeper level, say that I've created a pointer to ARR with
> IDL> PARR = ptr_new(ARR, /no_copy)
> Will freeing the pointer with
> IDL> ptr_free, PARR
> also release the temporary variable?
>
> 3. Is there a way to see how many temporary variables I have checked
> out? Is there a limit to that number? What's the limit?
>
> Many thanks,
>
> David

Following up on my earlier post, I have some additional insights
and a couple of questions for folks who really understand IDL internals.

As originally written, my program allocated a steady stream of temporary variables
using IDL_MakeTempArray() at a rate of 30 allocations per second. After about 20 minutes,
the whole system would lock up hard (no mouse or keyboard, no ssh). This behavior
was reproducible.

Just doing the math suggests that lock-ups occur after 2^15 allocations.

If, for the sake of argument, the number of allocated temporary variables were
maintained internally with a signed short integer, failures might arise when that
counter rolls over.

Attempting to release the temporary variable with methods 1 and 2 as I originally
proposed does not fix the problem. I even tried passing the temporary
array back to C to delete the variable (using IDL_DelTemp) , but the variable no
longer has its "temporary" flag set.

What works is to release the variable explicitly at the end of the same
C routine that creates it, just like in the documentation and example code.
This isn't useful, however, because I want to work with the variable in IDL.

This leaves me with a question: What is the "correct" way to allocate a temporary
variable in C, pass the variable back to IDL, and then free the variable's resources
in IDL?

Also, am I right about the upper limit on the number of temporary variables that can
be allocated in IDL?

All the best,

David

P.S. I've fixed my immediate problem by allocating my temporary variable
just once and feeding it back to my C library for updating (30 times per second).
The C code is messier, but seems to run reliably.
