
Subject: Re: How to ensure arrays are de-allocated in IDL?
Posted by [Jeremy Bailin](#) on Sat, 28 Aug 2010 14:57:55 GMT
[View Forum Message](#) <> [Reply to Message](#)

On Aug 28, 8:59 am, Robin Wilson <ro...@rtwilson.com> wrote:

> Hi,
>
> I've written some code that processes a large number of files, and
> creates around 5 large arrays during the processing of each of these files.
>
> However, after I get to around 200 files the program crashes saying that
> it can't find enough memory to create the arrays.
>
> Do I need to do something specific to ensure the memory for the arrays
> used in previous iterations are de-allocated?
>
> Regards,
>
> Robin Wilson
> University of Southampton www.rtwilson.com

Are they just held by normal variables, or allocated with pointers or in an object? If they're normal variables, then the memory should be reclaimed when you re-assign the variable name (for example, in the next iteration of the loop). You can force it to be reclaimed earlier by making the variable undefined, using, for example, David's UNDEFINE procedure (<http://www.dfanning.com/documents/programs.html>). If you're allocating them with pointers, then you'll need to explicitly free them with PTR_FREE, and with objects you'll need to destroy the object with OBJ_DESTROY (although with the garbage collection in IDL 8, the last case will become less relevant...).

-Jeremy.
