Subject: Use of IDL_MemAllocPerm() to prevent memory fragmentation?
Posted by Laurens on Mon, 19 Oct 2009 15:39:23 GMT
View Forum Message <> Reply to Message

Hi all,

I'm still fighting my little problem with memory allocation in IDL. I have a couple of interpolated CT scans loaded into memory (roughly one dataset for transversal, one set for coronal and one for sagittal, plus some overhead for seperate / unchanged administration). This can be seen as UINT arrays of [500,500,500].

Simply addressing them to variables quickly runs you into problems, so I'm using shared memory mapping, using the functions shmmap, shmvar and shmunmap. Works great....so far.

Just when I need to copy those arrays into a new variable in order to save them to new DICOM files, the heap is empty...at least, not enough continuous space is available to allocate that array.

I looked into switching to 64bit IDL (thanks to Benjamin Luethi), but too bad; that ver. of IDL (at least in version 6.3 - to which I'm bound) doesn't support the DICOM or DICOMex library. Well, I kinda need them in the first place to read the images.

So, I need better understanding of how to achieve my goal. Am I doing a strange thing here? In the manual I read that, quote, "IDL_MemAllocPerm() allocates memory in moderately large units and carves out pieces of these blocks to satisfy its requests. Use of this routine can help minimize the effects of memory fragmentation.". Is that a realistic option?

One other thing I don't understand; I'm using virtual memory for my objects. VM gets swapped to harddisk under windows. Shouldn't the memory I've used for past objects, be moved to the pagefile when I request a new block of memory? Eg. When I first load a block of 500x500x500x2bytes and then need a new block of the same dimensions, that first block should be moved to pagefile, removing the problem of not enough virtual mem being available? I guess I'm missing some crucial points here...

A lot has already been said about memory handling under IDL (mostly: accept the fact that on 32bit OS, more than 1.2GB of allocation is not going to happen and you ought to be lucky to find one piece of memory that big). But despite those comments, I'm still wondering if there are solutions I'm not seeing....

Hope someone has a clever day...