Subject: Re: memory
Posted by Karl Schultz on Wed, 30 Aug 2006 16:46:54 GMT
View Forum Message <> Reply to Message

On Wed, 30 Aug 2006 10:06:01 -0600, Jean H. wrote:

> Hello all,
>
> Is there a way for IDL to find out the amount of available memory?
>
> Also I am not too sure to understand how IDL is managing the virtual
> memory. In the help file, they say that IDL would automatically write
> information on the disk if the physical memory gets full. However, I
> have a function that crashes (i.e. % Unable to allocate memory.) as soon
> as my physical memory is totally used. In this function, I have to carry
> a few quite large arrays that are used only once in each loop
> iteration.. so technically speaking, they could be written on the disk
> instead of being kept in memory!
> I have 2G of physical memory, and the same amount of virtual memory (for
> a suposedly nice total of 4G then).
>
> IDL> print, !version
> { x86 Win32 Windows Microsoft Windows 6.2 Jun 20 2005 32 64}
>
> Any thoughts would be greatly appreciated!
> Jean
>
> PS: Having to add some loops in your code because the loopless one takes
> too much memory is more painfull than pulling your hairs when writing
> the "optimized" code..

This topic has been covered a great deal in this newsgroup over the years.
You might want to check the archives if this summary isn't enough.

1) 32-bit Windows reserves half of the 4G address space for the OS. I'm
not sure, but that may have improved to a 3G/1G app/os split in XP.

2) IDL does not manage its virtual storage. It lets the OS do it. As
physical memory gets committed, the OS will page out blocks of memory that
have not been used recently to disk.

3) Most memory allocation failures on Windows are due to virtual address
space fragmentation. (This is completely independent of paging) The
problem is that sometimes there is not enough free CONTIGUOUS virtual
address space to satisfy a request for a large allocation, even though
there are more than enough smaller free blocks around to fill the request.
You may find that you can't allocate a 500MB array, but you can allocate
4 or 5 200MB arrays.

These are all just characteristics of 32-bit Windows. You'll either have to redesign your algorithms to not rely on such large allocations or move to a 64-bit OS. A 64-bit address space suffers less from fragmentation.

Some folks have reported better success by moving to 32-bit Linux, because the virtual address fragmentation problem is not as severe. Windows divides up the user portion on the virtual address space into areas for specific uses and can load things in the middle of large free blocks. Both of these actions fragment the address space.

Karl
