Subject: Re: Where to find 64bit IDL 6.3?
Posted by Laurens on Mon, 19 Oct 2009 08:24:30 GMT
View Forum Message <> Reply to Message

On 16 okt, 19:34, Karl <karl.w.schu...@gmail.com> wrote:
> This topic has been covered here quite a bit.
>
> First, with 24GB of memory, you should really be running a 64-bit OS
> to take advantage of it. Only "server" additions of 32-bit Windows
> running apps that use PAE/AWE tricks to address past 32 bits can use
> more than 4GB.
>
> That being said, you should still be able to allocate 250,000,000
> bytes with 32-bit IDL on a 32-bit OS MAYBE.
>
> Here's why:
>
> 32-bit Windows (generally) maps the kernel stuff into the high 2GB of
> the 32-bit (4GB) address space, leaving user processes the low 2GB.
> Windows also uses up quite a bit of the lower 2GB of virtual address
> space to map shared DLL's and a number of other things. So, you can
> expect a user process to have access to 1-1.5 GB of virtual address
> space.
>
> Your array allocation requires contiguous virtual memory, so you need
> to sort of get lucky to find a contiguous block of the size you are
> requesting. Virtual memory can get fragmented over time and while
> there may be a total amount of free virtual address space exceeding
> your requirement, there may not be a free contiguous area that large.
>
> You can take some steps to improve your chances to find a big
> contiguous block:
>
> - Does your IDL application allocate other large arrays? If so, try
> allocating the big one first, before doing anything else.
> - Restart IDL between attempts in case your app has a leak that would
> fragment the address space.
> - Sometimes Windows can load DLL's that are shared between processes
> into the middle of a large block of free contiguous virtual address
> space, fragmenting it. Common culprits a few years ago were those
> applets that come with graphics cards that give you bells and whistles
> added to your windows' title bar. Might try disabling those. There
> are tools out there that can help you see what DLL's are mapped into a
> process and where. That can give you a clue as to what else might be
> fragmenting your memory.
> - See if you can use the "3GB" switch in your edition of Windows.
> This gives the kernel 1GB and the user 3GB of the virtual address
> space. Some people have had some luck with this.

>
> But even after all that, there is no guarantee of success.
>
> Going to a 64-bit OS/IDL solves the problem decisively by giving you a
> huge virtual address space, which is just as important as being able
> to use the RAM you are not using now.
>
> I'm not sure why you'd bother to put 24GB in a machine with a 32-bit
> OS unless running a server or special apps.
>
> -karl
>
> On Oct 16, 8:21 am, Paolo <pgri...@gmail.com> wrote:
>
>> On Oct 16, 5:45 am, Laurens <laur...@turboduif.nl> wrote:
>
>>> Hi Folks,
>
>>> I'm running in serious memory issues on 32bit windows version of IDL
>>> 6.3. Just trying to allocate an array of UINTS with dimensions
>>> 500x500x500 happens to be too much. I've already tried Memory Mapping
>>> by using shmmap and shmvar functions to let those arrays be written to
>>> harddisk, but without any luck, despite the system having 8 cpu's and
>>> 24GB of memory...
>
>> That's definitely a OS problem - in other OSes IDL 32 can
>> allocate a 500 cubed dblarr without problems. The rule is:
>> on IDL 32 bit the maximum size of an array is about 2^31
>> bytes (2GB). The max number of elements is about 2^31/S,
>> where S is the size of one elements in byte (2 for INT, 4 for
>> FLOAT etc.).
>
>> Ciao,
>> Paolo
>
>>> Now I'd like to try out the 64bit version of IDL. Since version 6.3,
>>> 64 bit should be present. The problem is that I can't find the
>>> installer anywhere...Does someone still have it around or can someone
>>> point me in the right direction here?
>
>>> Thanks in advance!
>
>>> Cheers, Laurens
>
>

In addition;
if I try to allocate that amount of memory, it will succeed initially.

I can do that command for four times before I get the "unable to allocate memory" message. Problem is, I need multiple variables in memory that big...at least I think. Chances are there is some leakage somewhere, but then again; how the hell do you find that in this amount of code? Any ways to get a listing of variables using up the memory?
