
Subject: Large TIFF file question

Posted by [Neil Talsania](#) on Tue, 15 Jan 2002 22:10:49 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hi,

I have what should be a simple question (I hope!). I am trying to run an IDL routine that was given to me. The routine has run successfully on small images, but when I try to run it on my 1.5 Gig image it fails on the memory allocation.

Looking at the code, it does the following:

```
a = float(read_tiff(filename)).
```

Now, I assume that means it is trying to read the entire thing into memory.

Ok I can see why that fails, although not really as I have >2G of RAM on my system. But anyway, I then thought about using ASSOC which according to the documentation might help with this problem. However, that requires me to open the file as a raw unformatted file (near as I can tell anyway).

So, to make a long story short, my question is how do I open a large TIFF file, without exhausting memory?

Thanks

Neil

Subject: Re: Large TIFF file question

Posted by [Dick Jackson](#) on Wed, 16 Jan 2002 17:24:48 GMT

[View Forum Message](#) <> [Reply to Message](#)

"Mark Rivers" <rivers@cars.uchicago.edu> wrote in message
news:dxg18.2\$s4.491@news.uchicago.edu...

>

> David Fanning <david@dfanning.com> wrote in message

> news:MPG.16af2ce16a9304e39897d3@news.frii.com...

>> Martin Downing (martin.downing@ntlworld.com) writes:

>>

>>> If you are crazy/unfortunate enough to be doing this on a windows OS,

>>> you'll be facing the 1/2Gb limit on process memory, [...]

>>

>> I thought one of the features of IDL 5.4 or 5.5 (I

>> can't recall, since I just woke up and I'm sitting

>> here scratching myself and waiting for the coffee

>> to boil) was an RSI hack that allowed the PCs to

>> exceed these memory limits. I remember this as being

>> one of the most significant, but completely unheralded,

>> items of that release.

>
> If you find anything documenting that I'd be most interested to hear about
> it. I routinely bump into this limit on Windows machines with 1GB of RAM,
> reading 3-D tomography data sets that are 400-600 MB. .RESET_SESSION_ALL
> sometimes helps, but I have to exit/restart IDL very frequently because
the
> memory gets fragmented.

It seems that 5.4 and 5.5 on Windows 2000 will allow me to make a half-GB
(512 MB) array, does this (dis-)prove anything? FYI, I have exactly 512MB of
RAM, VM set somewhat higher.

=====

IDL Version 5.5, Microsoft Windows (Win32 x86). (c) 2001, Research Systems,
Inc.

```
IDL> a=bytarr(1024,1024,512)
IDL> help,/mem
heap memory used: 537264198, max: 537264226, gets: 436, frees: 211
IDL> a[1023,1023,511]=42
IDL> print,a[1023,1023,510:511]
0
```

42

=====

IDL Version 5.4 (Win32 x86). (c) 2000, Research Systems, Inc.

```
IDL> a=bytarr(1024,1024,512)
IDL> help,/mem
heap memory used: 537245837, max: 537245865, gets: 433, frees: 200
IDL> a[1023,1023,511]=42
IDL> print,a[1023,1023,510:511]
0
```

42

=====

Cheers,

--

-Dick

Dick Jackson / dick@d-jackson.com
D-Jackson Software Consulting / http://www.d-jackson.com
Calgary, Alberta, Canada / +1-403-242-7398 / Fax: 241-7392

Subject: Re: Large TIFF file question

Posted by [Martin Downing](#) on Thu, 17 Jan 2002 12:03:16 GMT

[View Forum Message](#) <> [Reply to Message](#)

> "Mark Rivers" <rivers@cars.uchicago.edu> wrote in message

> news:dxg18.2\$s4.491@news.uchicago.edu...

>>

>> David Fanning <david@dfanning.com> wrote in message

>> news:MPG.16af2ce16a9304e39897d3@news.frii.com...

>>> Martin Downing (martin.downing@ntlworld.com) writes:

>>>

>>>> If you are crazy/unfortunate enough to be doing this on a windows OS,

>>>> you'll be facing the 1/2Gb limit on process memory, [...]

>>>

>>> I thought one of the features of IDL 5.4 or 5.5 (I

>>> can't recall, since I just woke up and I'm sitting

>>> here scratching myself and waiting for the coffee

>>> to boil) was an RSI hack that allowed the PCs to

>>> exceed these memory limits. I remember this as being

>>> one of the most significant, but completely unheralded,

>>> items of that release.

>>

I'd be very interested to find out more - anyone know where this is reported?

>> If you find anything documenting that I'd be most interested to hear about

>> it. I routinely bump into this limit on Windows machines with 1GB of RAM,

>> reading 3-D tomography data sets that are 400-600 MB.

.RESET_SESSION_ALL

>> sometimes helps, but I have to exit/restart IDL very frequently because

> the

>> memory gets fragmented.

>

> It seems that 5.4 and 5.5 on Windows 2000 will allow me to make a half-GB

> (512 MB) array, does this (dis-)prove anything? FYI, I have exactly 512MB

of

> RAM, VM set somewhat higher.

>

Allocating memory for a large variable is one thing, but you may not be able to use it effectively as it will most likely be paged out by windows. There is a command in windows 2000 (VirtualLock in Win32Api) which allows a program to hold onto physical RAM (i.e. stop paging) but I do not think RSI have considered this option. I believe the feature David is referring to is "large file support"[1], this is an issue of being able to read write to

files that are over 2.1GB and so need a 64bit (long64) file pointer.
However, I dont think it would be a good idea to try to load one of these files fully into memory!!!
On memory, the same documentation implies 32 bit IDL should be able to handle upto 2.1Gb of data, according to Microsoft[2], if you have standard NT/2000 then your processes can address 2Gb, with NT Server you can address 3GB and with NT Enterprise edition 4GB.

Martin

[1]"Large File Support for Windows platforms"
whatsnew.pdf (IDL5.4) pages 31-32

[2]Very Large Memory Partitions:

http://www.microsoft.com/ntserver/ProductInfo/Comparisons/UN IX/NTappdev/4_En tSupportFeat.asp

Subject: Re: Large TIFF file question
Posted by [karl_schultz](#) on Thu, 17 Jan 2002 20:18:21 GMT
[View Forum Message](#) <> [Reply to Message](#)

"Martin Downing" <martin.downing@ntlworld.com> wrote in message news:<[d4z18.37791\\$_x4.5640421@news2-win.server.ntlworld.com](mailto:d4z18.37791$_x4.5640421@news2-win.server.ntlworld.com)>...
>> "Mark Rivers" <rivers@cars.uchicago.edu> wrote in message
>> news:<[dxg18.2\\$s4.491@news.uchicago.edu](mailto:dxg18.2$s4.491@news.uchicago.edu)...
>>>
>>> David Fanning <david@dfanning.com> wrote in message
>>> news:<MPG.16af2ce16a9304e39897d3@news.frii.com...
>>>> Martin Downing (martin.downing@ntlworld.com) writes:
>>>>
>>>> > If you are crazy/unfortunate enough to be doing this on a windows
> OS,
>>>> > you'll be facing the 1/2Gb limit on process memory, [...]
>>>>
>>>> I thought one of the features of IDL 5.4 or 5.5 (I
>>>> can't recall, since I just woke up and I'm sitting
>>>> here scratching myself and waiting for the coffee
>>>> to boil) was an RSI hack that allowed the PCs to
>>>> exceed these memory limits. I remember this as being
>>>> one of the most significant, but completely unheralded,
>>>> items of that release.
>>>>
>>>
>
> I'd be very interested to find out more - anyone know where this is
> reported?

I don't remember this sort of thing. I believe that IDL on Windows was always using the memory system as much as the OS would let it. I do remember that whenever we looked into memory issues for customers, the problem usually boiled down to an OS limitation. I'll check around and post again if I discover anything different.

I also know that in IDL 5.2, IDL on all platforms supported the theoretical 32-bit addressing maximum.

There's a good chance that David's pre-coffee memory may have been thinking of file I/O limits.

Limits on process memory do vary among Windows releases, which might account for some of the confusion. For example, NT4 splits the 4GB virtual address space into two, leaving 2G for apps and 2G for the OS (kernel mode). So, there would be a 2G limit on NT4, and not 4G.

```
>>> If you find anything documenting that I'd be most interested to hear
> about
>>> it. I routinely bump into this limit on Windows machines with 1GB of
> RAM,
>>> reading 3-D tomography data sets that are 400-600 MB.
> .RESET_SESSION_ALL
>>> sometimes helps, but I have to exit/restart IDL very frequently because
> the
>>> memory gets fragmented.
```

This may be caused by the application asking for a huge (400-600 MB) contiguous virtual memory space. This may be pretty hard to find, even in a 1 or 2 GB virtual space after running the program a few times. If the app had a way of using smaller chunks to store this data, it may suffer less from this fragmentation problem.

```
>>
>> It seems that 5.4 and 5.5 on Windows 2000 will allow me to make a half-GB
>> (512 MB) array, does this (dis-)prove anything? FYI, I have exactly 512MB
> of
>> RAM, VM set somewhat higher.
>>
>
> Allocating memory for a large variable is one thing, but you may not be able
> to use it effectively as it will most likely be paged out by windows. There
> is a command in windows 2000 (VirtualLock in Win32Api) which allows a
> program to hold onto physical RAM (i.e. stop paging) but I do not think RSI
> have considered this option.
```

I don't think that locking memory would be very good. The MS docs say:

(start quote from MS dev lib)

Locking pages into memory may degrade the performance of the system by reducing the available RAM and forcing the system to swap out other critical pages to the paging file. By default, a process can lock a maximum of 30 pages. The default limit is intentionally small to avoid severe performance degradation. Applications that need to lock larger numbers of pages must first call the `SetProcessWorkingSetSize` function to increase their minimum and maximum working set sizes. The maximum number of pages that a process can lock is equal to the number of pages in its minimum working set minus a small overhead.

(end quote)

I think that this API call is intended for device drivers that really, really, really need to have memory pages locked into real memory for real-time performance reasons. If we went ahead and locked a bunch of memory, there would be fewer pages for other processes running in the OS, causing crashes and a LOT of paging as these "must-run" processes must complete for a small set of unlocked pages. It would be a real mess. It just isn't polite for an application to do this sort of thing. It makes the other applications running on the system and the user very unhappy.

To help IDL use more RAM:

- Add more RAM :-)
- Reduce the number of other applications running at the same time. The chances are good that a lot of the pages used by these apps will get swapped out as they become inactive, but many will get or stay paged in to some extent. I kill other apps whenever I want a single app to run as fast as possible on my machine.
- Organize the application to reduce its working set size. The TIFF sub-image suggestion already mentioned here is actually the best solution/approach for the original problem. If an app allocates a lot of storage and then walks around all over and through it, there's going to be a lot of paging. To be honest, on Windows, the paging subsystem is so slow (compared to a lot of UNIX systems) that a lot of customers I know split their data up into chunks so that it all fits into RAM. Other approaches include performing large array operations along the appropriate dimensions in order to keep all the array accesses in the same area as much as possible. If you must page, then getting a faster disk I/O system may help.

Anyway, you can approximate the same effect of locking pages by

reducing the competition for those pages using any of these three approaches. Just let the OS do its job. The stale pages will eventually page out and IDL will use more and more of the memory pages as other apps give them up.

- > I believe the feature David is referring to is
- > "large file support"[1], this is an issue of being able to read write to
- > files that are over 2.1GB and so need a 64bit (long64) file pointer.
- > However, I dont think it would be a good idea to try to load one of these
- > files fully into memory!!!

Nope.

- > On memory, the same documentation implies 32 bit IDL should be able to
- > handle upto 2.1Gb of data, according to Microsoft[2], if you have standard
- > NT/2000 then your processes can address 2Gb, with NT Server you can address
- > 3GB and with NT Enterprise edition 4GB.

I didn't know about the larger user process space in the Server and Enterprise editions. If that's true, then you should be able to allocate past 2G with these systems, but that is my untested guess. The documentation you refer to may have had plain NT4 in mind.

- >
- > Martin
- >
- > [1]"Large File Support for Windows platforms"
- > whatsnew.pdf (IDL5.4) pages 31-32
- >
- > [2]Very Large Memory Partitions:
- >
- > [http://www.microsoft.com/ntserver/ProductInfo/Comparisons/UN IX/NTappdev/4_En](http://www.microsoft.com/ntserver/ProductInfo/Comparisons/UN IX/NTappdev/4_En tSupportFeat.asp)
- > tSupportFeat.asp

Hope this helps,
Karl
