

Re: memory leak`s

Source: <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.mfc/2005-04/msg00020.html>

- *From:* Joseph M. Newcomer <newcomer@xxxxxxxxxxxxx>
 - *Date:* Thu, 31 Mar 2005 12:09:08 -0500
-

Yes, the task manager is a first approximation if you know how to interpret it, but relying on it for an instantaneous snapshot whose meaning is not interpreted sensibly is misleading. In my essay, for example, I point out what you said. A consistent monotonic increase is a serious indicator, although whether it represents a memory leak or increasing memory fragmentation is not determinable just by watching the memory increase. Looking at the behavior when the program is minimized is largely meaningless.

joe

On Sun, 20 Mar 2005 20:37:07 -0800, "Alexander Grigoriev" <alegr@xxxxxxxxxxxxx> wrote:

>If committed virtual memory is steadily increasing, there is likely a memory
>leak. Since the lost memory may not be accessed anymore, the working set may
>stay the same, but the program may eventually run out of the virtual memory.
>
>Task Manager may show USER objects count increasing, too, if GDI resoures
>are leaked.
>
>I guess the MFC debug memory allocator may show the lost allocations at the
>program shutdown.
>
>"Joseph M. Newcomer" <newcomer@xxxxxxxxxxxxx> wrote in message
>news:3qoo31ppjlp3v3udnnfiug256mv4f9lgng@xxxxxxxxxxx
>> Fascinating.
>>
>> First, it is incorrect to use 0xffffffff as a value. The documentation
>> states
>> (SIZE_T)-1
>> and that is important, because on a 64-bit processor, it will set the
>> working set size to
>> 4GB (0xffffffff is not (SIZE_T)-1 on a 64-bit processor).
>>
>> Second it is clear that your clients are idiots. It is more important for
>> them to see a
>> small size than to see acceptable performance! By doing this, you swap the
>> entire program
>> out of memory, which then means you take page faults like crazy.
>>
>> If you get an L1 cache hit, you get 0 CPU cycles delay in accessing the
>> data

Re: memory leak`s

>> If you get an L1 cache miss, but an L2 cache hit, you lose 1 or 2 CPU
>> cycles
>> If you get an L2 cache miss, you lose 20 CPU cycles
>> If you take a page fault, you lose 20,000,000 CPU cycles
>>
>> (Note: this data may be obsolete. You may be losing far more than
>> 20,000,000 CPU cycles
>> these days!). So at best, you are forcing a six-orders-of-magnitude
>> performance penalty on
>> your program for each page fault it has to take!
>>
>> I would spend more time on customer education.
>> joe
>>
>> On Thu, 17 Mar 2005 05:14:32 +0100, Gert
>> <onlyReplyInNewsgroups@xxxxxxxxxxx> wrote:
>>
>>> Dave wrote:
>>>> Hi,
>>>> I am writing an mfc vs-6 application, when I am running my application
>>>> for a
>>>> long time I see in the task manager that the virtual memory is
>>>> increasing,
>>>> but the regular memory is the same I look in my application to find
>>>> memory
>>>> leak`s but I didn't find any even when I use third party application
>>>> they
>>>> didn't find any memory leak`s!
>>>> How can I find it and fix it???
>>>> why only the virtual memory is increasing???
>>>>
>>>> Thanks!
>>>>
>>>>
>>>>
>>>> Try minimizing your app while you look at task-manager. You will see the
>>>> memory use drop a lot. I got some customers complaining about memory use
>>>> showing on task-manager, so now my program calls
>>>> SetProcessWorkingSetSize(GetCurrentProcess(), 0xffffffff, 0xffffffff);
>>>> every few minutes. I guess this is what windows does when it minimizes
>>>> your application.
>>>>
>>>> Gert
>>
>> Joseph M. Newcomer [MVP]
>> email: newcomer@xxxxxxxxxxx
>> Web: <http://www.flounder.com>
>> MVP Tips: http://www.flounder.com/mvp_tips.htm
>

Joseph M. Newcomer [MVP]

Re: memory leak`s

Re: memory leak`s

email: newcomer@xxxxxxxxxxxxx

Web: <http://www.flounder.com>

MVP Tips: http://www.flounder.com/mvp_tips.htm

.

-
- Prev by Date: ***Re: menu in MDI***
 - Next by Date: ***Re: Need an STL (std:string) API call example***
 - Previous by thread: ***Re: Handling OnActivateApp()***
 - Next by thread: ***Re: memory leak`s***
 - Index(es):
 - ◆ ***Date***
 - ◆ ***Thread***