

Re: MDAC memory leak

Source: <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.mfc/2008-05/msg00405.html>

- *From:* Joseph M. Newcomer <newcomer@xxxxxxxxxxxxx>
 - *Date:* Fri, 09 May 2008 21:31:45 -0400
-

When allocations are done, particularly from libraries, there always arises the issue of when the storage should be freed. Most libraries place the decision of when to free storage on the caller. So if you comment out a call to a library, and storage leaks go down, you need to investigate under what circumstances the storage is freed. It could well be that you are failing to notify the library that you are finished with the last set of data before initiating a call to obtain new data. Since I don't know what libraries you are using, I can't offer advice, but in libraries I've written, and third-party libraries I've used, there is typically a requirement to either delete the objects returned or to call the library to delete the objects returned.

strcore leaks are hard to diagnose, because they mean you have an unfreed CString. This is actually fairly rare in most cases, and what you most often see is a CString as a member of some class leaking because the instance of the object is still hanging around. There's a capability of breaking on a particular memory allocation, so if the effect is reproducible in that sequences of dumps always give you a leak for allocation { 1024 } then you set the memory allocator breakpoint value to 1024, and you will see where it is called. You will probably find a CString inside a CSomething that is a member of an array or list of a CWhatever, and the real problem is not that you are leaking a string, but that you neglected to free the CWhatever object when you were done with it. I tend to leave strcore.cpp leaks as the very, very, very last leaks I will ever look for. When I've fixed the other leaks (and it is usually a systematic pattern; on the first test, I leak 500 objects, on the second test I leak 3, because I fixed the bug). The new smart pointers also reduce memory leakage by making it impossible to leave dangling objects around.

Oh, yes, the other cause is that you DID delete the CWhatever, but the CWhatever had a CSomething* member, and you didn't delete the CSomething* in the ~CWhatever destructor. That's usually where I go awry, and it is usually a stupid oversight on my part (I MEANT to put the delete in, but got distracted in some way and forgot about it...)

So first look for major leaks, and worry about strcore leaks as the very, very last thing, and you may not have to deal with them at all.

It helps if you take a look at the line that says

```
#define new DEBUG_NEW
```

or whatever it is (including its #if... conditionals) and make sure all your modules have it. This sometimes helps, because it forces the debug allocator call to be issued, which

Re: MDAC memory leak

includes the file/line information in the object.
joe

On Fri, 9 May 2008 14:36:00 -0700, Prashant <Prashant@xxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote:

Joseph,

Thanks for you reply. Actually when we commented all the calls related to the database the memory usage as well as the amount of leak came down subsequently.

Also when we used some of the memory leak tools suggested on microsoft site those are showing leak in strcore.cpp.

Thanks,
Prashant

"Joseph M. Newcomer" wrote:

See below...

On Wed, 7 May 2008 02:11:00 -0700, App shows memory leak on some machines.

<Appshowsmemoryleakonsomemachines@xxxxxxxxxxxxxxxxxxxxxxxxxxxx>
wrote:

Actually I've originally used AfxBeginThread() function to launch thread and just a return statement while exiting thread. And since I was getting handle leak along with memory leak, I tried using CloseHandle() in the previously mentioned way. After this change I found no difference in app's behavior. I again revert back to original code where I've used a return statement in the top level thread function. Still I can find same behavior (handle + memory leak).

I'll like to add some more points here--

--I am launching only one thread from main thread.
Lets say ThreadFirst is launched by main thread. Now, all other threads are getting launched by ThreadFirst.
It's required since main thread needs to do other task.

Re: MDAC memory leak

–Again there is a critical section (semaphore) in thread function. Many threads are keep on hanging at the start of critical section waiting for others threads to leave.

I suspect you are confused. A CRITICAL_SECTION is a very efficient low-level synchronization primitive. A mutex is a considerably less efficient synchronization primitive. A semaphore is something completely different, and fulfills the purposes of a semaphore (a counted exclusion object) and has little to do with mutual exclusion of multiple threads. So when you say "critical section (semaphore)" I become suspicious, because semaphores would not be appropriate here.

And yes, if you have a synchronization object, and you have high contention, you are likely to have a lot of threads hanging.

Some rules:

Don't use synchronization. Design your code so there is no need for concurrent access

Lock data, not code

Lock the smallest possible amount of data for the shortest possible time

In the absence of any code example showing the locking, it is hard to guess what you might have done, but the most likely cause is that you have violated one or more of the above rules.

Locking is done when threads rub together. That means there is friction. And just like in mechanical systems, friction generates heat and wastes energy. I tend to view synchronization as something that should be avoided by making sure threads are not actually rubbing. See my essay "The Best Synchronization is No Synchronization" on my MVP Tips site. Every once in a while, you need concurrent access, but the more I work with threads, the less I do it.

joe

Re: MDAC memory leak

Can it create any problem??

Thanks,
Digvijay

"Nick Schultz" wrote:

Use Process Explorer from
www.sysinternals.com . It is free and is
MUCH
better than Task Manager, you can double
click on a process and get very
detailed information, including a list of
threads and memory usage. It also
shows processes in a tree format rather than
just a flat list...overall its
a better tool.

Go to the Options menu and select "Replace
Task Manager" and Process
Explorer will pop up whenever you press
CTRL+ALT+DEL.

Nick

"App shows memory leak on some
machines."

<Appshowsmemoryleakonsomemachines@xxxxxxxxxxxxxxxxxxxxxxxxxxxx>
wrote in
message
news:91870733-F651-4E90-A5E7-88CB6170E89D@xxxxxxxxxxxxxxxxxxxx

Thank you Joseph for your
reply.

I've used AfxBeginThread()
function to launch thread
and subsequently
AfxEndThread() to end it.
I tried using CloseHandle()
and checked its return value,
also checked
return value for
GetLastError(), to make
sure that handle is getting
closed.
Return values were as

Re: MDAC memory leak

expected. But, it made no difference in behaviour of exe.

This part of code looks like

```
//-----  
CWinThread* ThPtr =  
AfxBeginThread(..);  
bool b =  
::CloseHandle(ThPtr->m_hThread);  
int n = GetLastError();  
  
thread function:  
ThreadFunc()  
{  
.  
.  
.  
AfxEndThread();  
return bRet;  
}  
//-----
```

I've used socket and DB calls inside thread function. For DB communication, I've used 'msado15.dll'.

I'll like to know, in which cases application will behave differently (in terms of memory leaks) on different machine?

Thanks,
Digvijay

"Joseph M. Newcomer"
wrote:

It is FAR more likely that what happened is that you are failing to close the thread handle of

Re: MDAC memory leak

the thread,
resulting in
a leakage of
thread
stacks. Are
you
certain
every
thread
handle is
being
closed?

The task
manager
may suggest
that
memory
usage is
increasing
(it is one
of the few
valid
reports
about
memory it
is actually
capable of),
but you
need to rule
out
lots of other
explanations
before
pointing to
MDAC (not
that it might
not be the
cause,
but the most
common
error I've
found in
multithreaded
leaks is
stack
leakage). I
would
suggest
downloading
the free App

Re: MDAC memory leak

Verifier
from the
Microsoft
site and
turning on
all memory
management
options.
joe

On Fri, 2
May 2008
07:48:02
-0700,
Prashant
<Prashant@xxxxxxxxxxxxxxxxxxxxxxxxxxxx>
wrote:

The
application
is
a
multithreaded
exe
involving
socket
communication.
The
memory
usage
in
task
manager
increases
when
exe
is
running.
It
never
comes
down.
In
other
words,
for
each
thread
it
increases

Re: MDAC memory leak

and
never
comes
down
even
after
that
thread
dies.

Thanks,
Digvijay

"Joseph
M.
Newcomer"
wrote:

How
do
you
detect
that
there
is
a
memory
leak?
You
assert
this
is
happening
without
explaining
why
you
think
it
is
so.
joe
On
Thu,
1
May
2008
05:11:01
-0700,
App

Re: MDAC memory leak

shows
memory
leak
on
some
machines.
<App
shows
memory
leak
on
some
machines.@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx>
wrote:

Hi,

My
application
(developed
in
VC++)
is
running
fine
on
most
of
the
machines(without
any
memory
leak).
But,
it
is
showing
memory
leaks
on
production
machine.
Even
on
development
side
it
is
showing
memory

Re: MDAC memory leak

leaks
on
few
machine.
Why
it
is
so?

Hardware
configuration
of
these
machines
are
different.
But,
I
think
this
difference
should
not
create
any
memory
leak
in
application.

My
application
is
using
database
communication
and
socket
communication
APIs.

Is
there
any
relation
bet
Hardware
configuration
and
MDAC
version????

Re: MDAC memory leak

Is
MDAC
2.82.3959.0
having
memory
leak
problems??

I've
checked
my
code
several
times
with
different
APIs
and
tools,
but
it
has
no
memory
leak.
Please
help
me
solving
the
problem.
I've
tried
to
solve
problem
in
many
ways
for
a
long
time,
but
not
getting
any
output.

MDAC
versions

Re: MDAC memory leak

on
these
machines
are
mentioned
below.
Development
machine
:
2.81.1128.0
Production
machines
:
2.82.3959.0

OS
:
XP
SP2,
Windows
2003
server

Joseph
M.
Newcomer
[MVP]
email:
newcomer@xxxxxxxxxxxxx
Web:
<http://www.flounder.com>
MVP
Tips:
http://www.flounder.com/mvp_tips.htm

Joseph M.
Newcomer
[MVP]
email:
newcomer@xxxxxxxxxxxxx
Web:
<http://www.flounder.com>
MVP Tips:
http://www.flounder.com/mvp_tips.htm

Re: MDAC memory leak

Joseph M. Newcomer [MVP]
email: newcomer@xxxxxxxxxxxxx
Web: <http://www.flounder.com>
MVP Tips: http://www.flounder.com/mvp_tips.htm

Joseph M. Newcomer [MVP]
email: newcomer@xxxxxxxxxxxxx
Web: <http://www.flounder.com>
MVP Tips: http://www.flounder.com/mvp_tips.htm