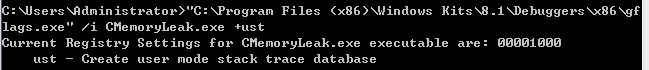
## Windbg to detect memory leak

* First of all, you need to install windbg obviously.
* Then use gflags.exe to create user mode stack trace database. Gflags.exe is also a windows debug tool and you can get it from the same path of windbg.

Execute following command:

Gflags.exe /i <\*\*\*.exe> +ust

Sample:



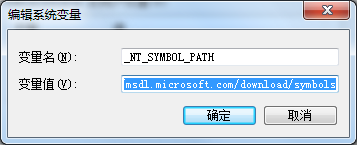
* Set the symbol file path for windbg

Note: The symbol files must be ***full PDB files***. Virtual Studio has different choice to generate pdb. For VS2017, it’s in Project Property -> Configuration Properties -> Linker -> Debugging. Need to set ***Generate Debug Info***to ***/DEBUG:FASTLINK*** and set ***Generate Full Program Database File*** to ***Yes.*** This doesn’t make much sense to me. This issue is submitted in Microsoft community, but this issue is not recognized as a bug.

<https://developercommunity.visualstudio.com/content/problem/104034/debug-generates-partial-pdb.html>

Add environment variable for the windows symbol files path, so windbg will load it automatically.

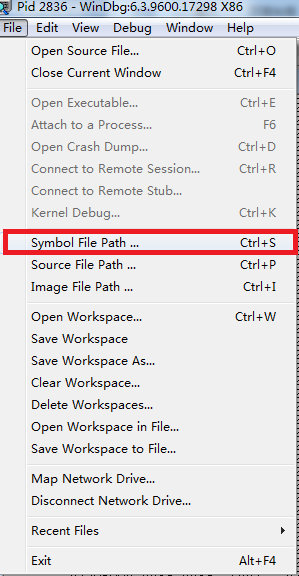
\_NT\_SYMBOL\_PATH = SRV\*c:\symbols\*http://msdl.microsoft.com/download/symbols

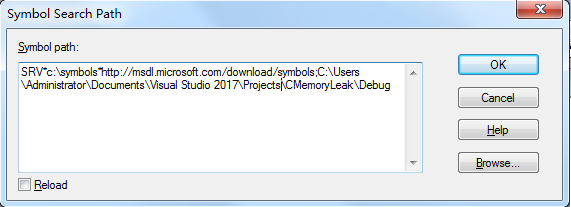


GUI: File->Symbol File Path(use ; to split multiple symbol file path)

Command: .sympath+ <Path>

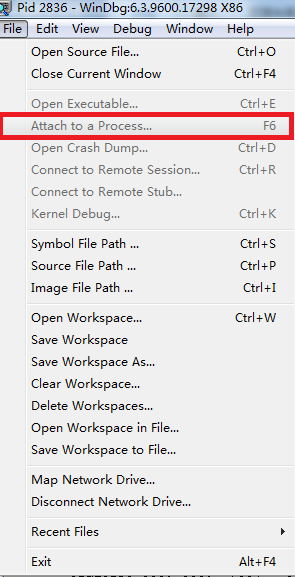
Ref: <https://blog.csdn.net/ecjtu_luowei/article/details/43974727>



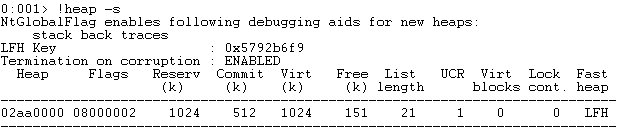


* Execute the application
* Attach to the process

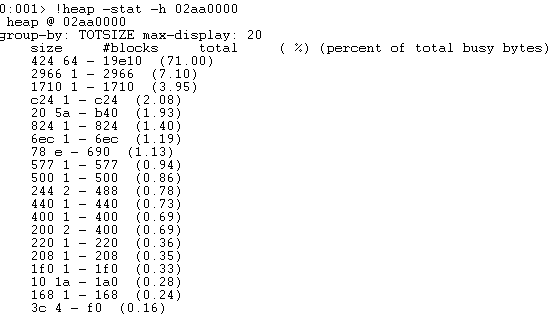
Note: after attaching the process, the process will stop on the point you attached. Need to use command ‘g’ in windbg, if you want it to continue run.



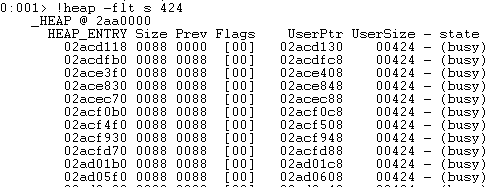
* Check the heap with command ‘!heap’
* Display a heap summary with ‘!heap -s’



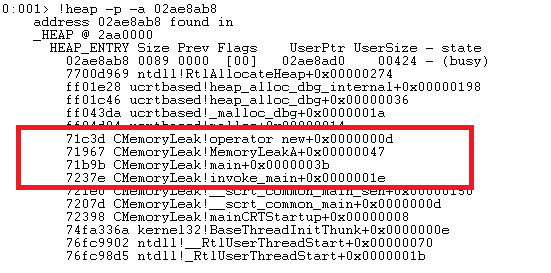
* Show the memory blocks summary for a specific address with ‘!heap -stat -h <address>’



* Filter out the address whose block size is the same with memory block that takes most of the memory. ‘!heap -flt s <size>’



* Show the memory stack of heap. ‘!heap -p -a <address>’



* Then analyse the stack you will found the place leads to the memory leak.

Reference: <https://www.cnblogs.com/SkyMouse/archive/2012/07/05/2578553.html>

## Windbg to analyze un-handled exception

Set the **Full** PDB to Windbg and access to the process before the exception. Then continue to run the process until the exception occurred. Windbg will show where the exception happened in and even the source code.

And type command ***k*** to show the call stack.

