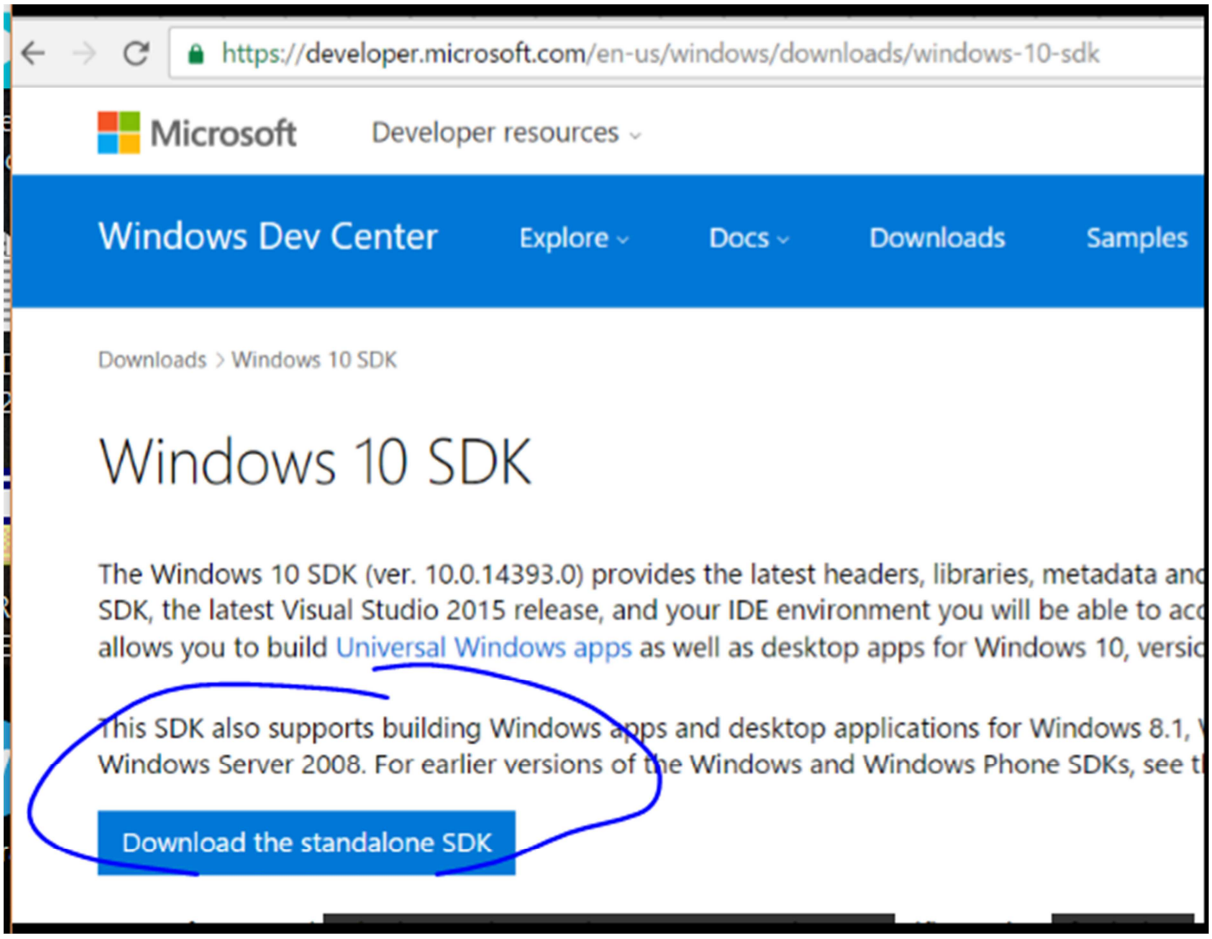
PART 32

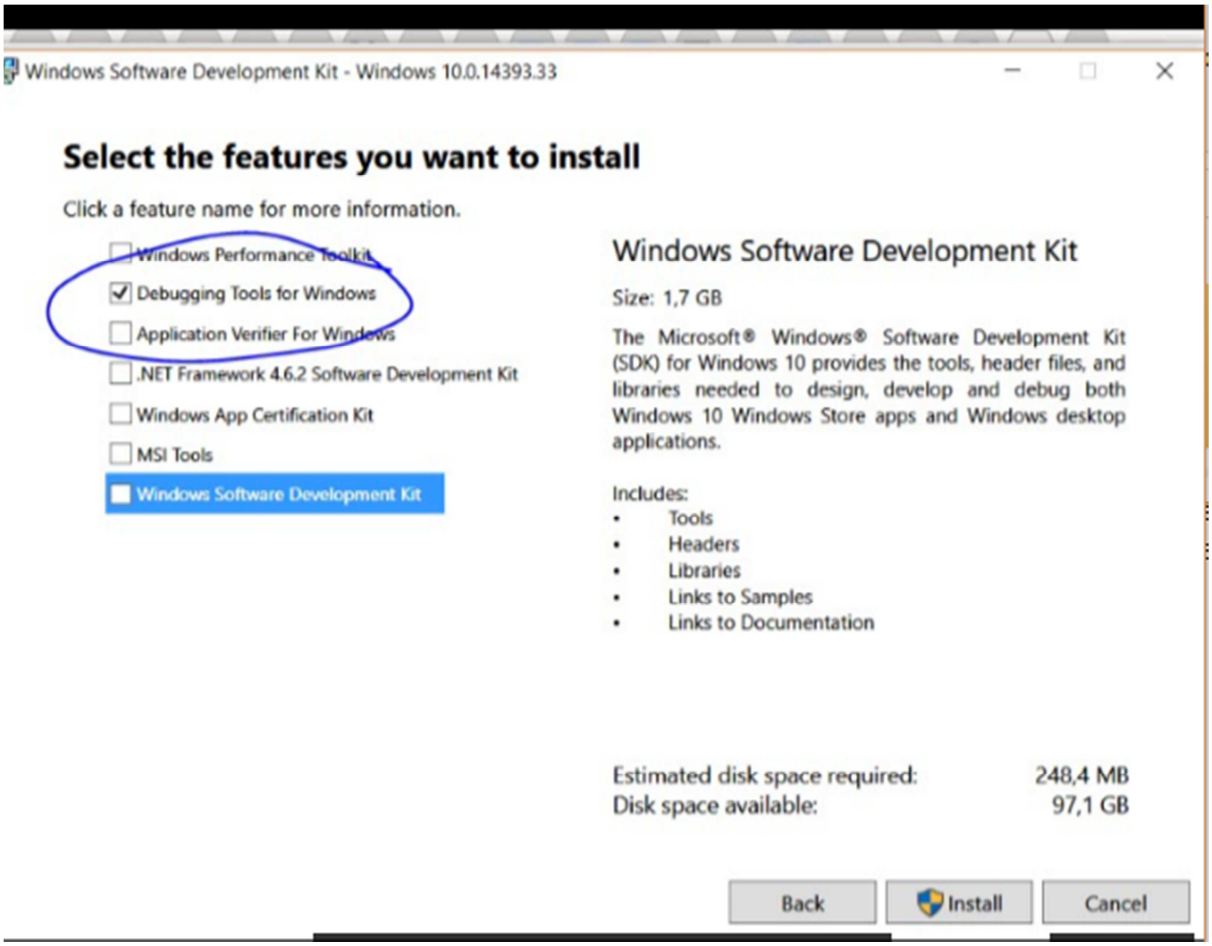
Preparing Windbg to work with IDA.

We need to install Windbg. It can be handled through IDA interface. It is a great debugger. Maybe, it lacks comfort because it has console-type commands and it is a little complicated to use it, but we can use the IDA great interface debugging with the Windbg engine. It is useful when we want to debug kernel or have good info about the heap status in heap overflow or after free bugs that we will see later.

There are many ways to install WinDbg. Unfortunately, they vary and you'll have to try the ones that work in your PC. As I am using Windows 10, I downloaded Windows 10 SDK from Microsoft website.

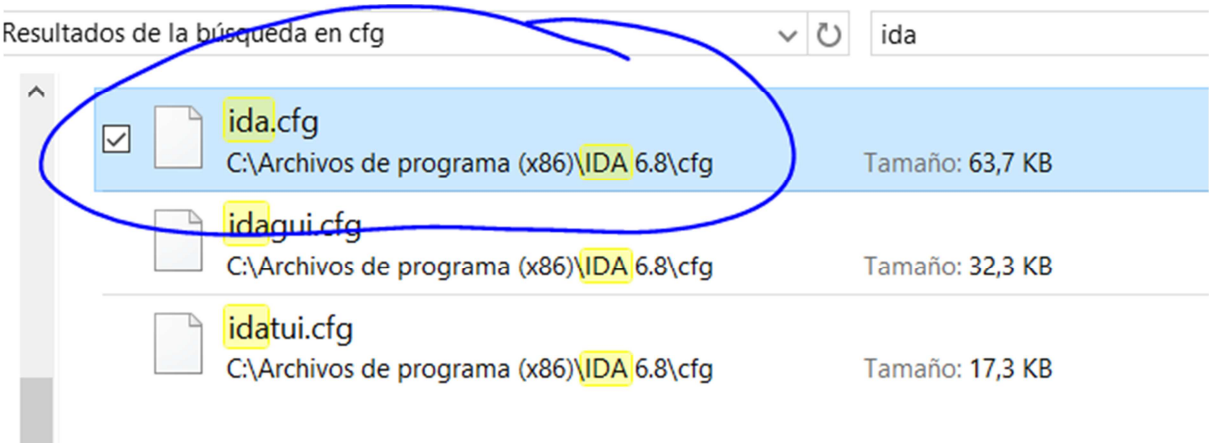


When the options appeared, I just selected DEBUGGING TOOLS FOR WINDOWS.

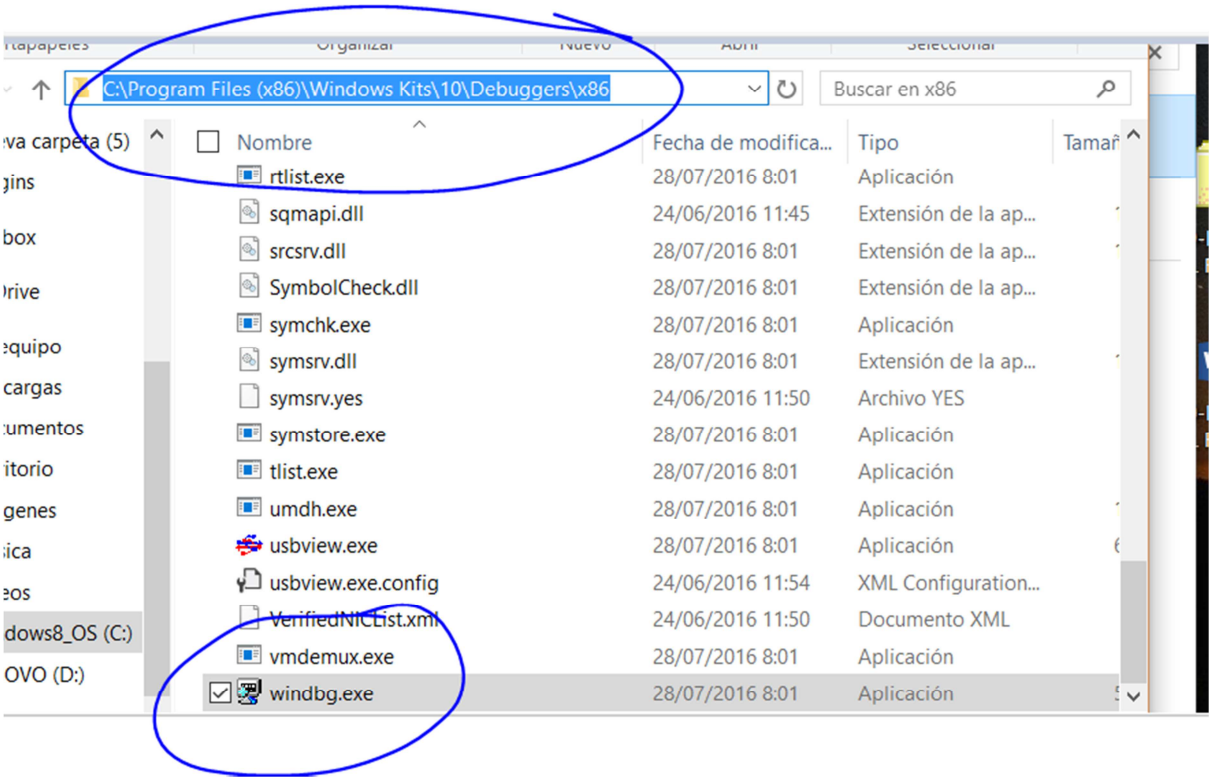


Unchecking the other options, it will install WinDbg. As each Windows has its own SDK, you could do the same in other Windows and install the correspondent one in your PC. At least, in this case, we know it goes well.

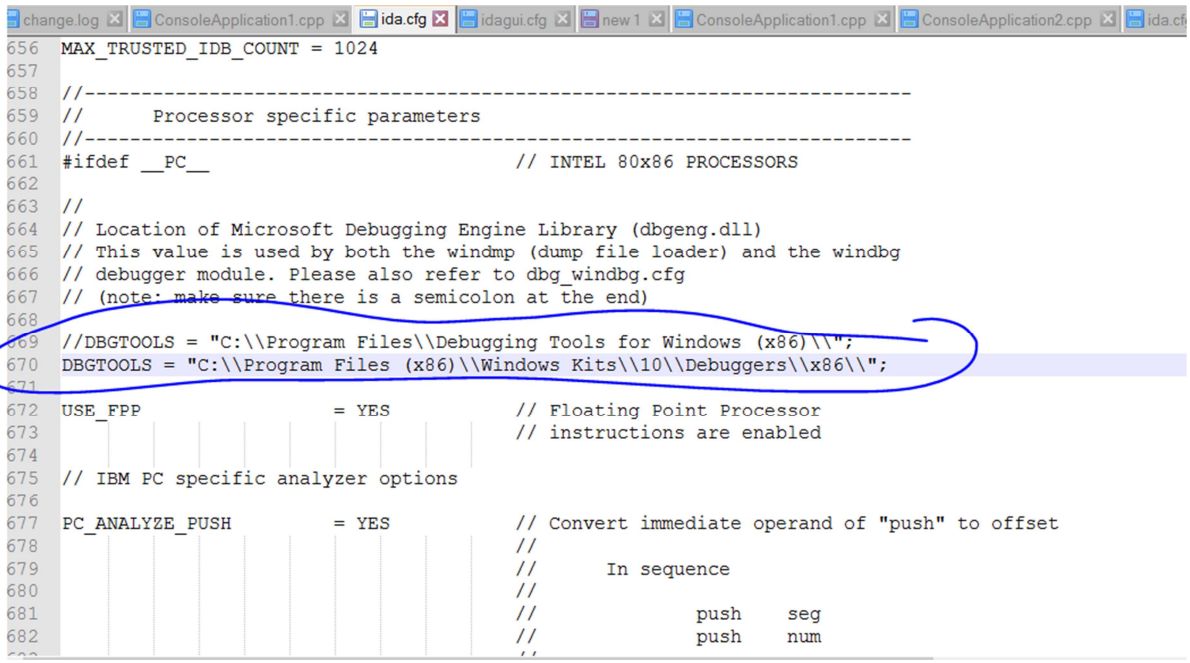
Then, I have to go to the cfg folder inside IDA installation and look for IDA.CFG



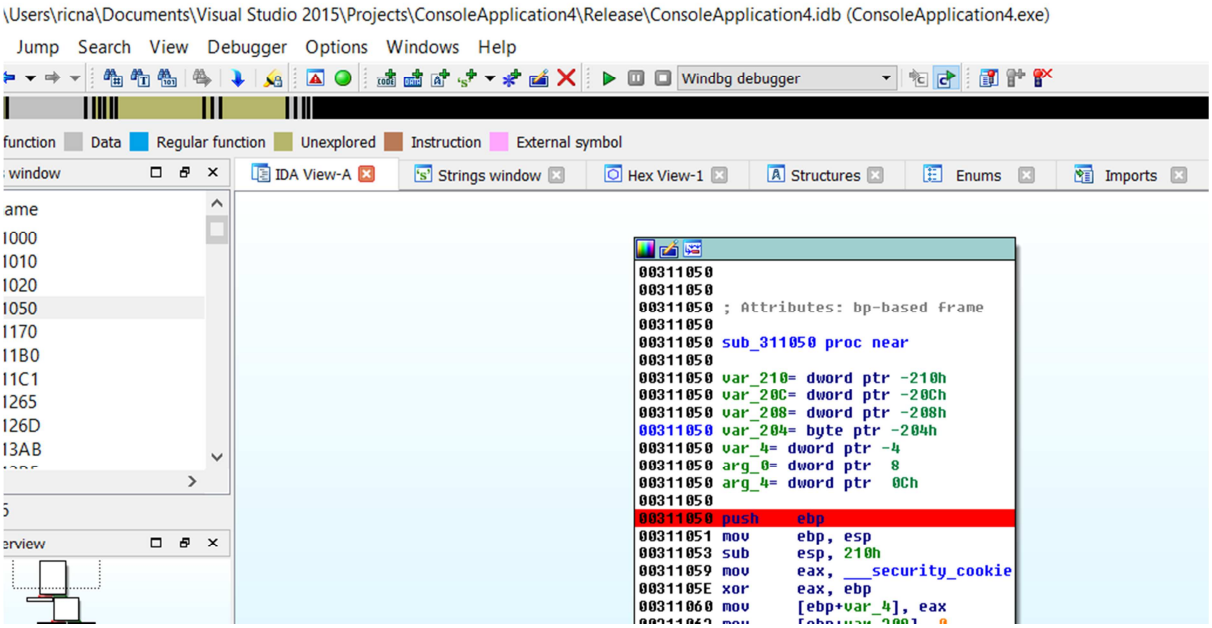
I edit that file and I add the path in DBGTOOLS where WinDbg x86 is installed. In my case, it is in:



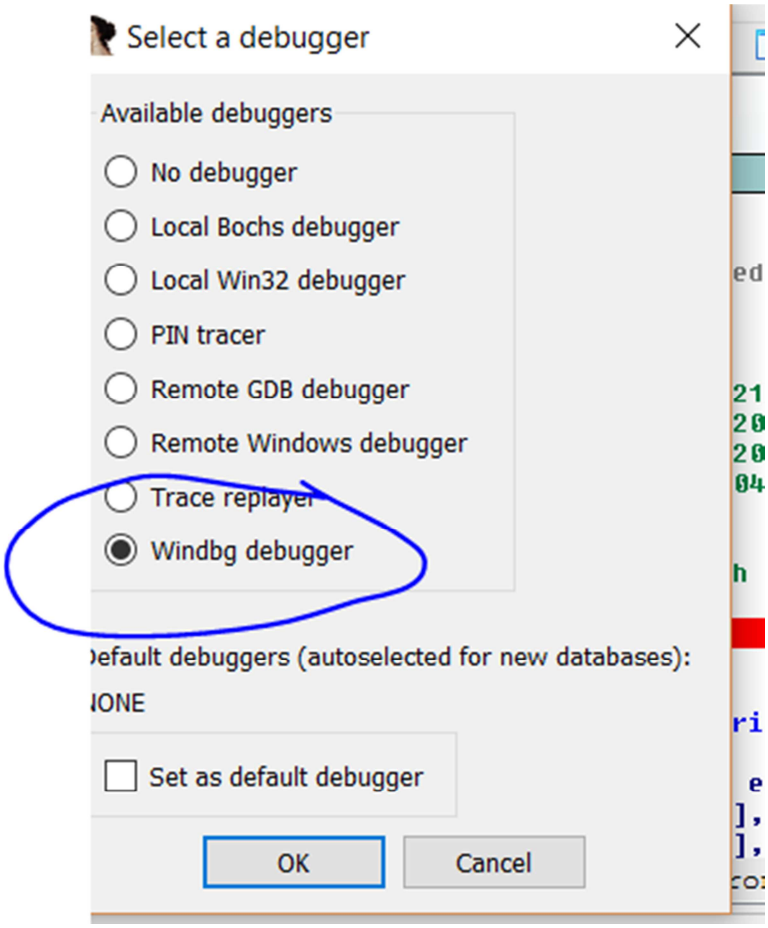
So, in the IDA.CFG file, I find DBGTOOLS.



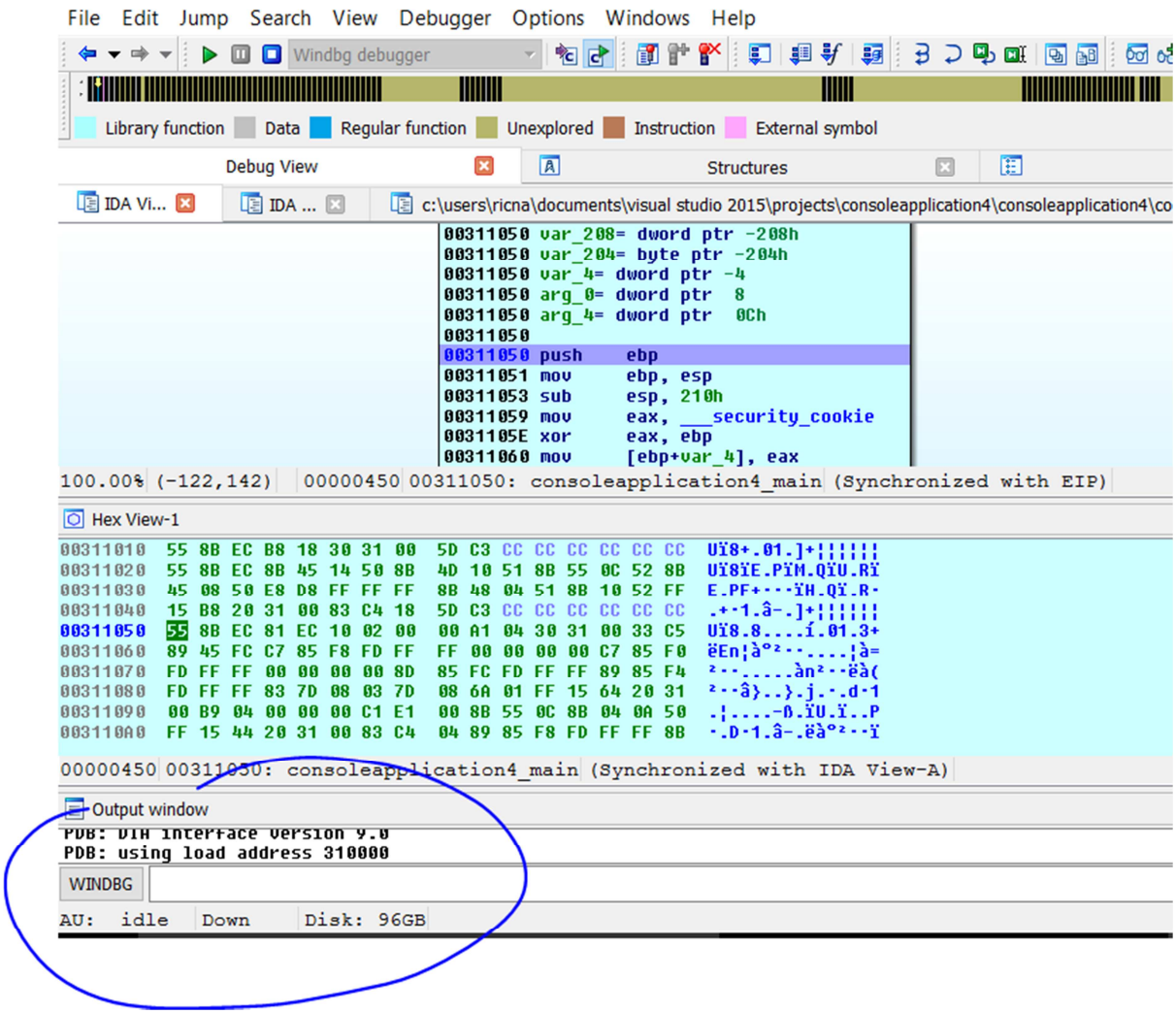
And I add the exact path adding the double bar \\ instead of the single one to separate the folders. I left the original path commented above.



I open any executable and set a BreakPoint to stop and I change the debugger to WinDbg.

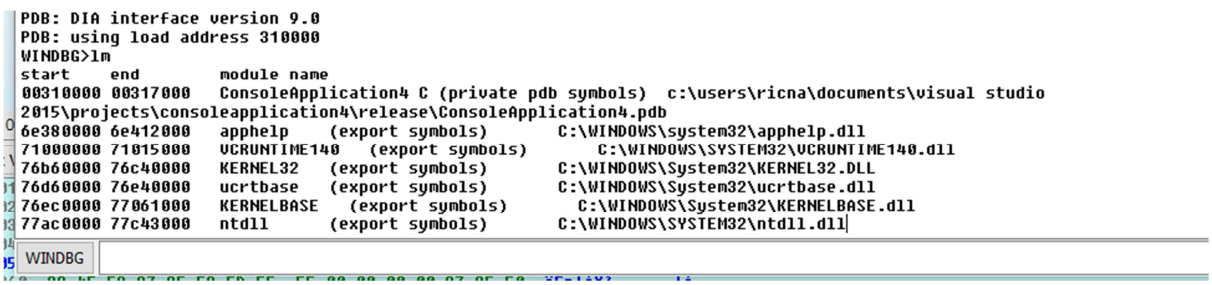


Then, we'll check if it was OK. We run the executable. If it says it doesn't find WinDbg, you should check the path or something failed in the installation.



There, it stopped as it were IDA Win32 local debugger, but we see that below where the Python bar is normally, it says WindDbg. Clicking on the word WinDbg, I can switch to the Python bar.

This means everything is fine until now. We can use IDA GUI and WinDbg commands. Let's try some of them.



lm (List Loaded Modules)

The **lm** command displays the specified loaded modules. The output includes the status and the path of the module.

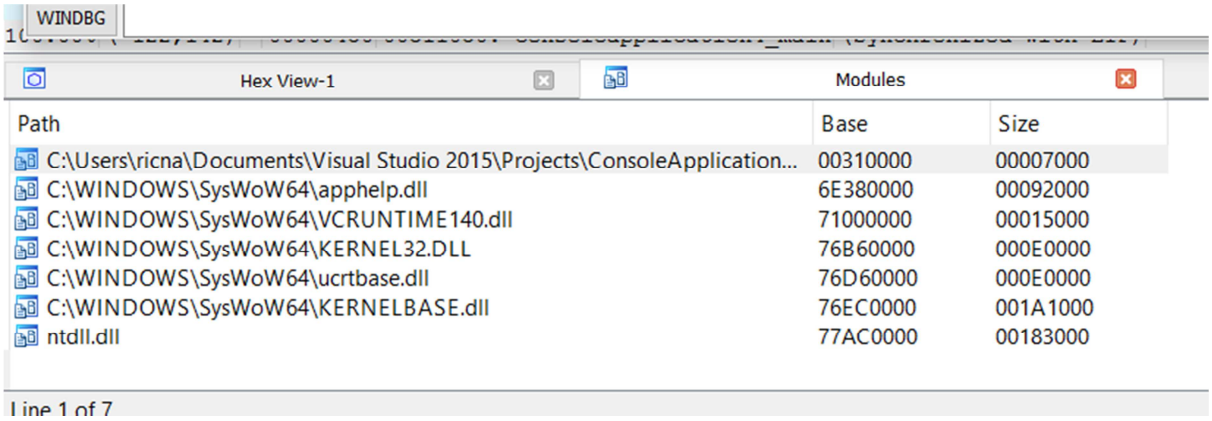
```
lmOptions [a Address] [m Pattern | M Pattern]
```

Parameters

Options

Any combination of the following options:

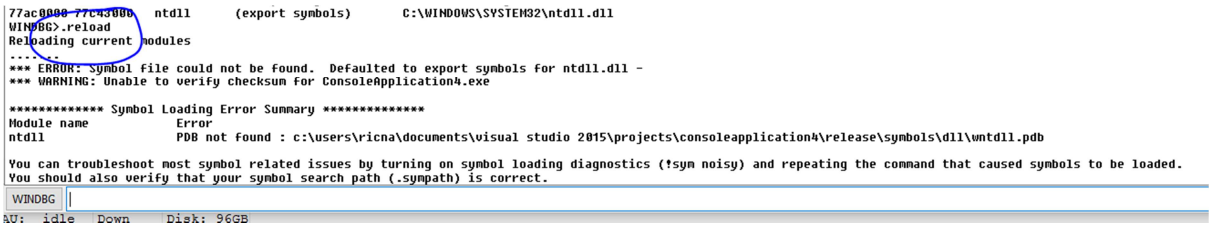
It worked. I can see the module list in Windbg bar. Obviously, in IDA too.



We also have to set the symbols for WinDbg. If we type…

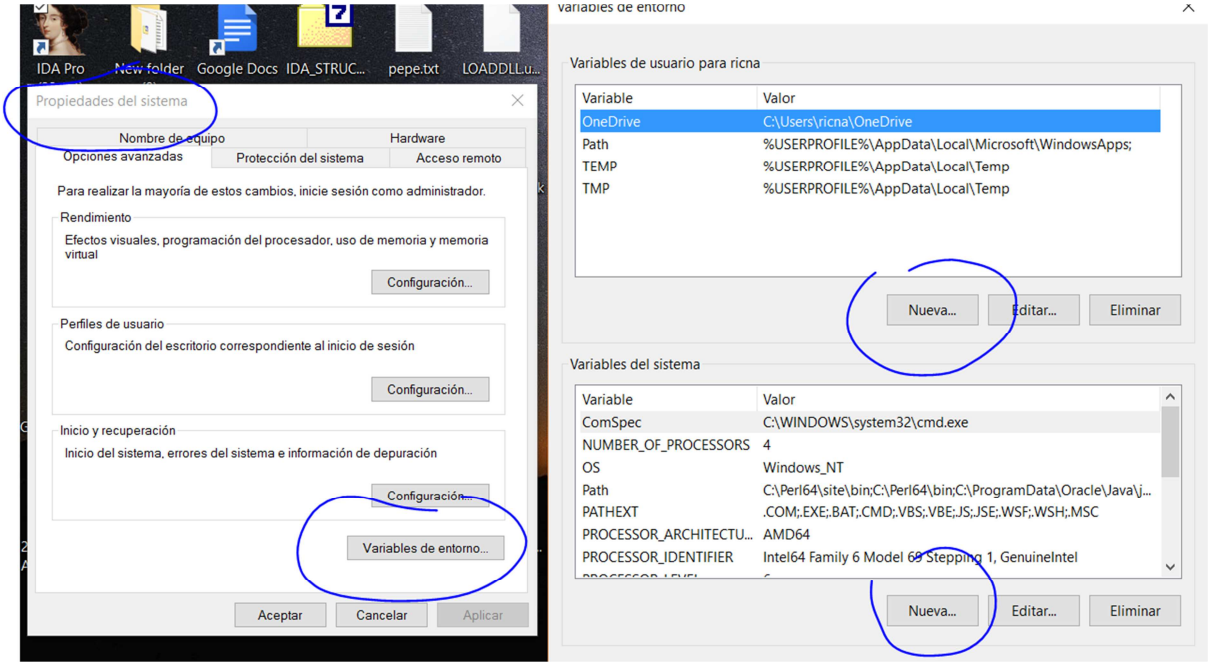
.reload

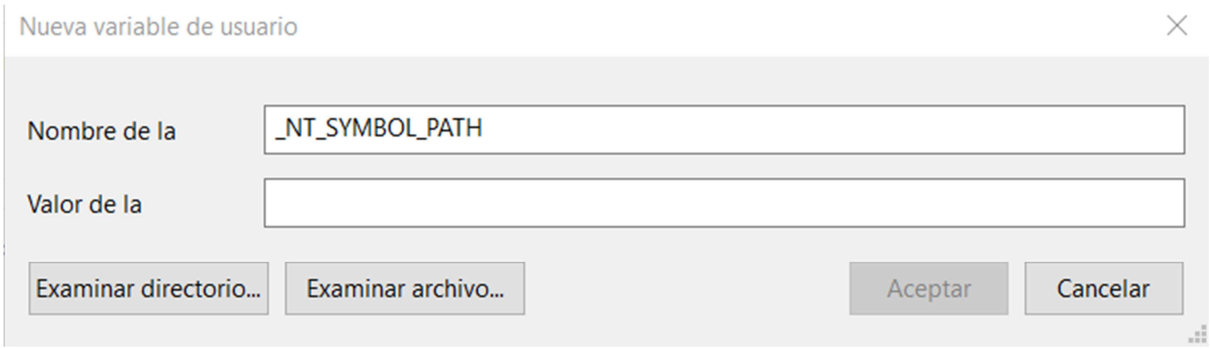
It shows error with symbols when they are not set where to save them.



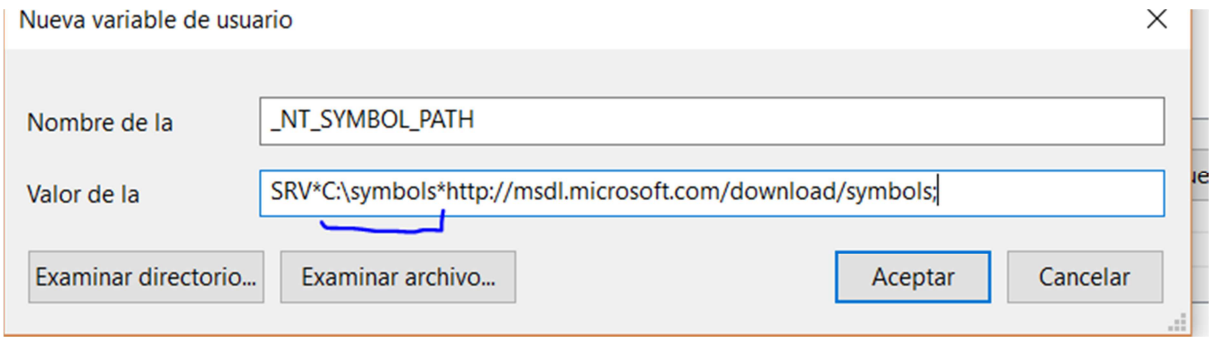
Create a folder in C for the symbols. IDA will have to run as administrator. If not, it won't write there.

In Windows ENVIRONMENT VARIABLES, we add _NT_SYMBOL_PATH.



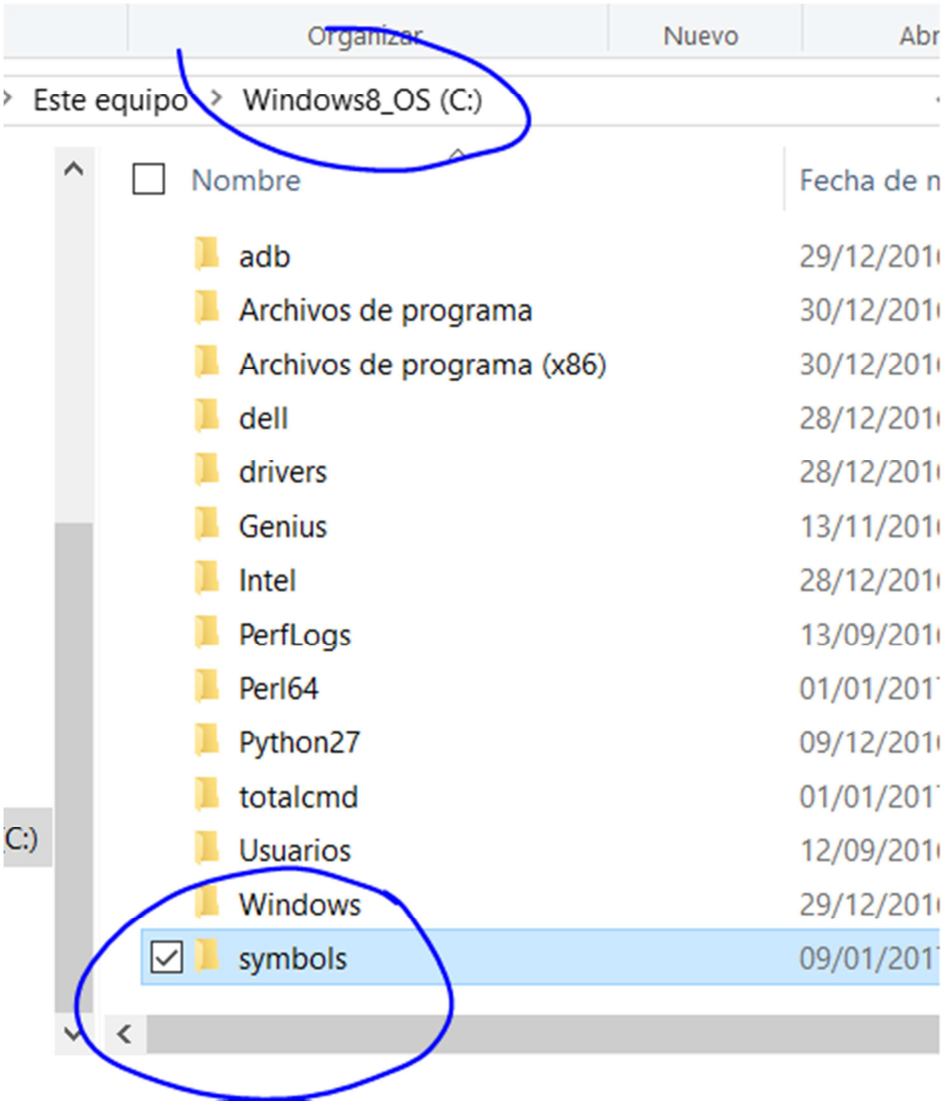


And in values, we type the following:



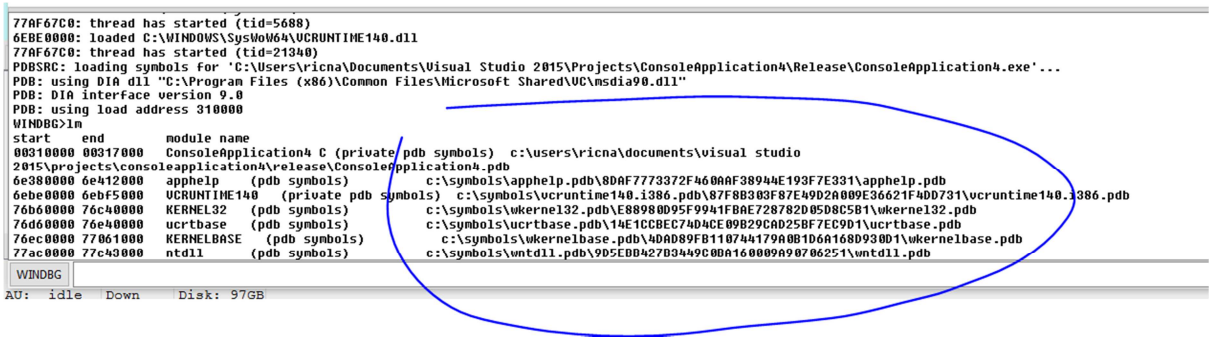
There, I will type the folder path where the symbols will be, in my case, it will be the **symbols** folder in C.

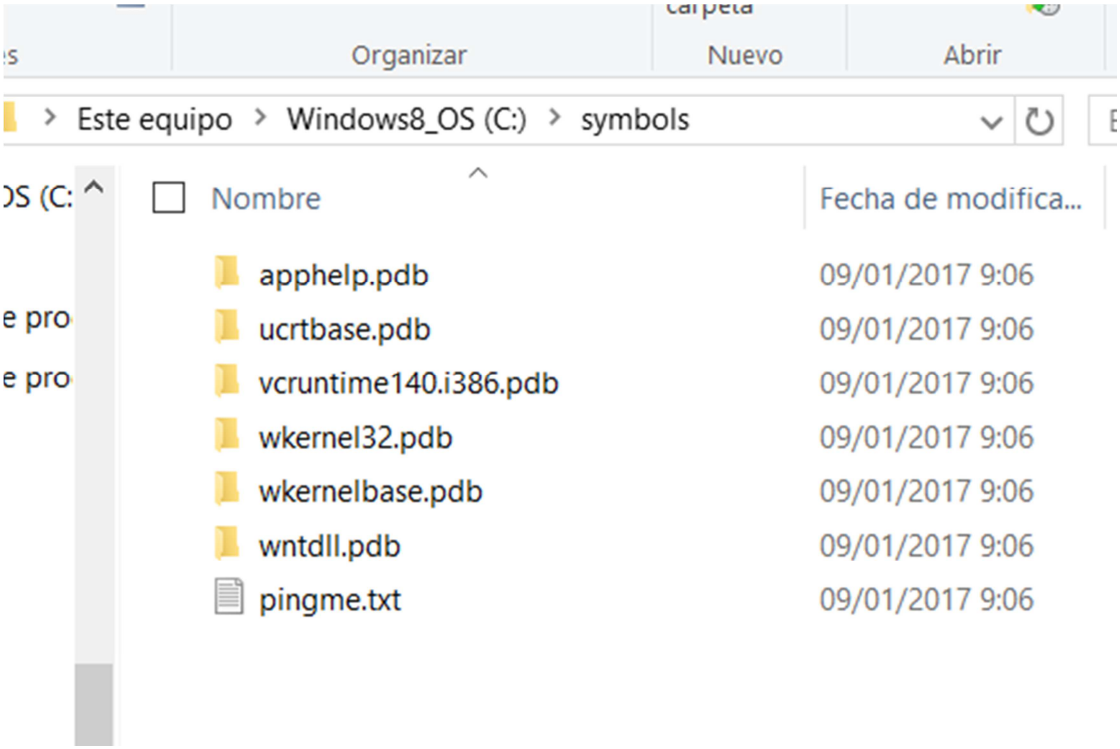
Alli pondre el path a la carpeta que cree donde se bajara los símbolos, en mi caso sera la carpeta symbols en C.



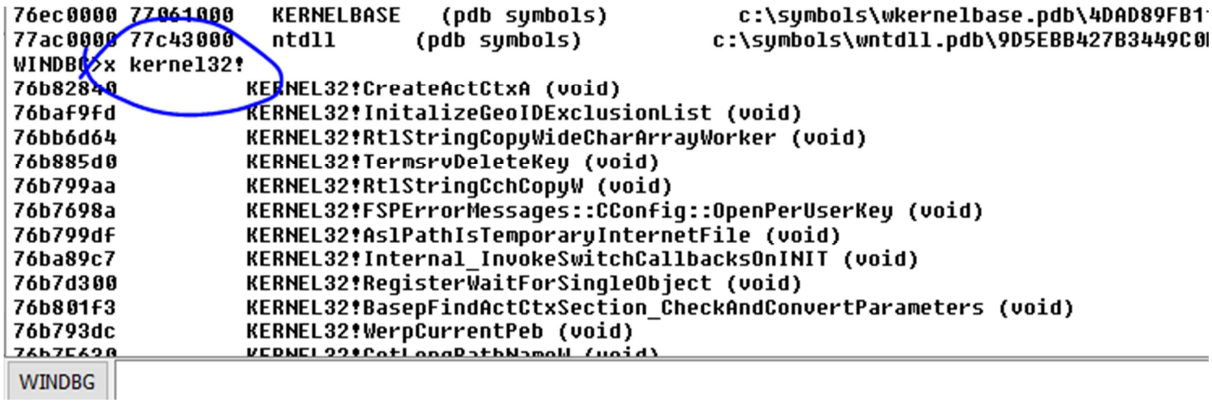
I restart my PC or the Windows explorer (explorer.exe killing and running it from the process bar)

When I run IDA again and run the debugger with WinDbg, I see it tries to download the symbols and when it finishes with lm, the path we set appears and the PDBs are inside the **symbols** folder.

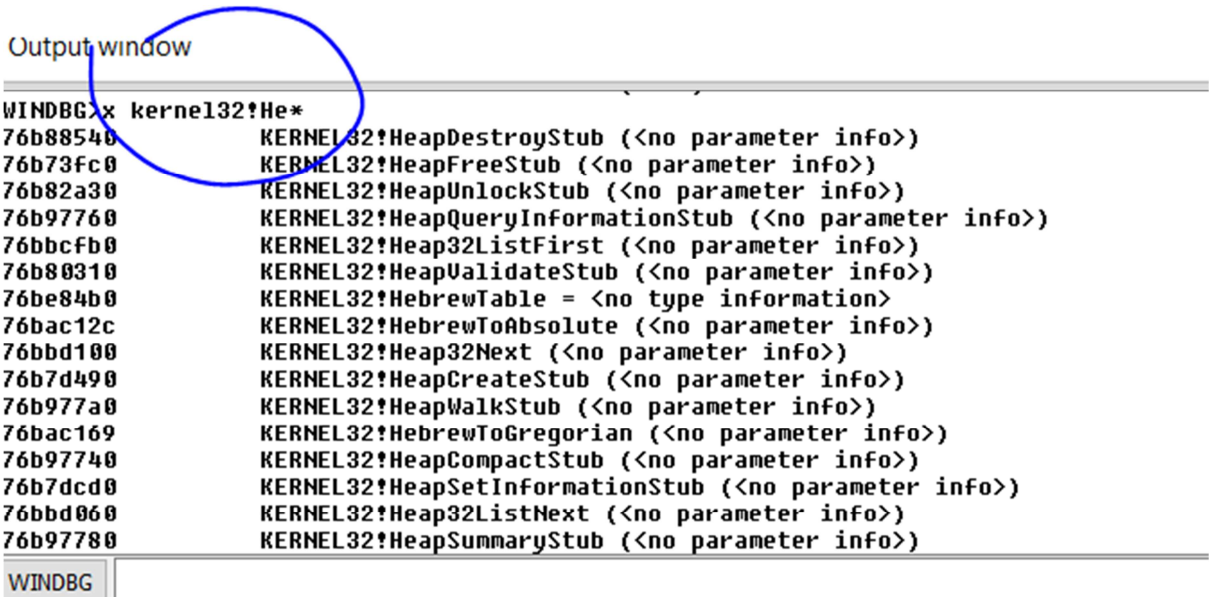




It is ready to work.

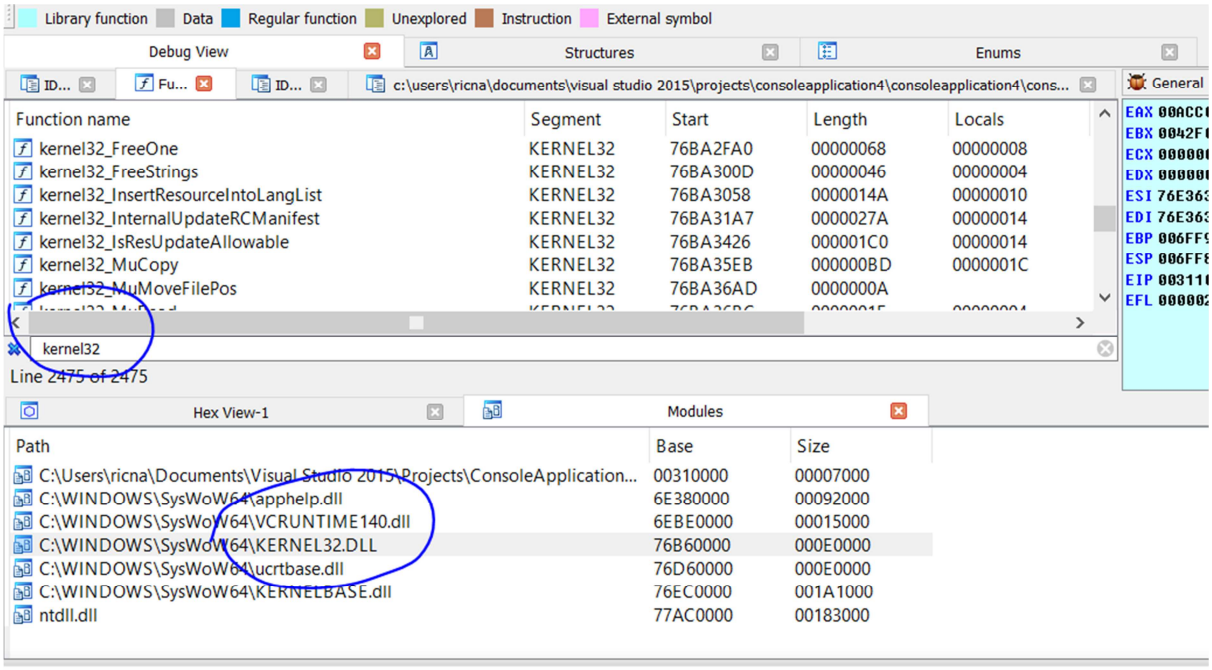


We can see the kernel32 function list. We can also use wildcards.



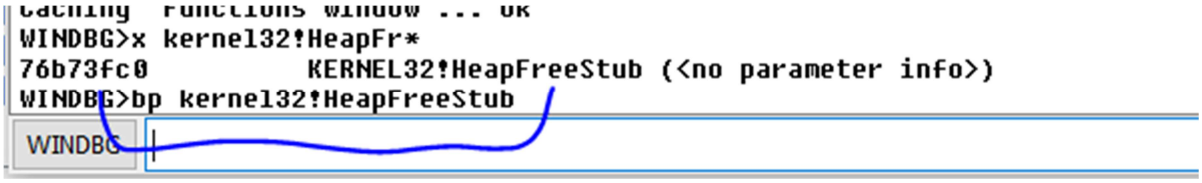
The ones that start with **He**.

I can also right click - LOAD SYMBOLS on any modules in IDA

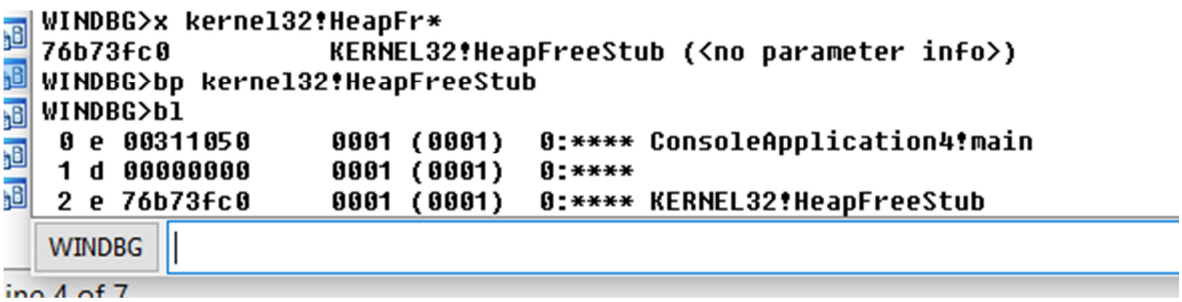


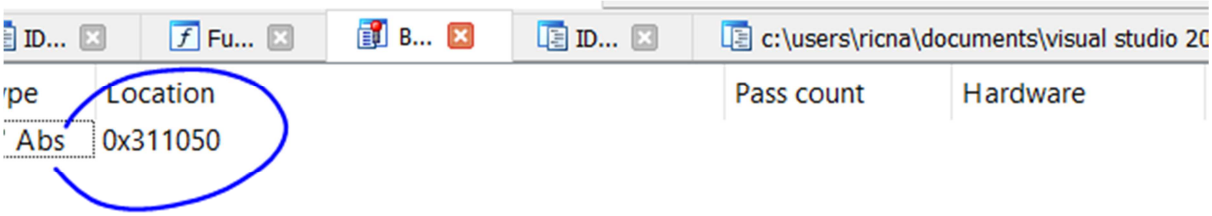
I will also have them in IDA interface.

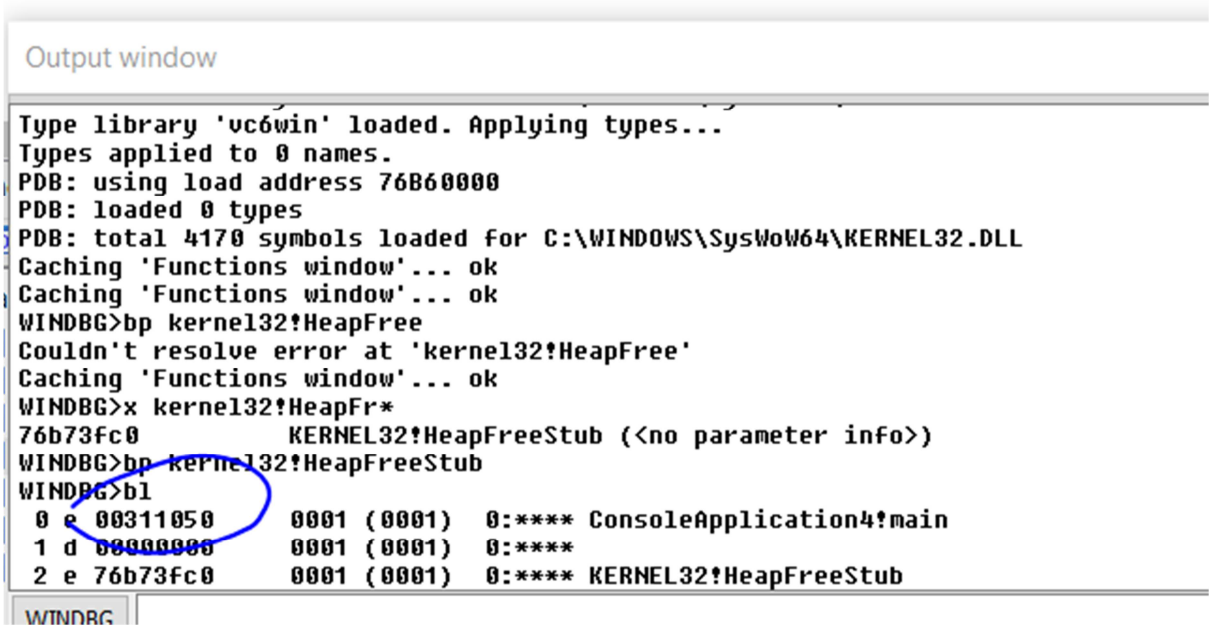
I can set breakpoints in IDA interface as we do it normally with F2 and through WinDbg bar that will be handled by it.



I type **BL** to list WinDbg breakpoints.







In WinDbg, all BPs are listed while in IDA, it only shows IDA BPs, but it will always stop at all them.

It is important that you prepare the installation to make it work well. Sometimes, there are problems with that. So, I will continue in the next part.

Ricardo Narvaja

Translated by: @IvinsonCLS