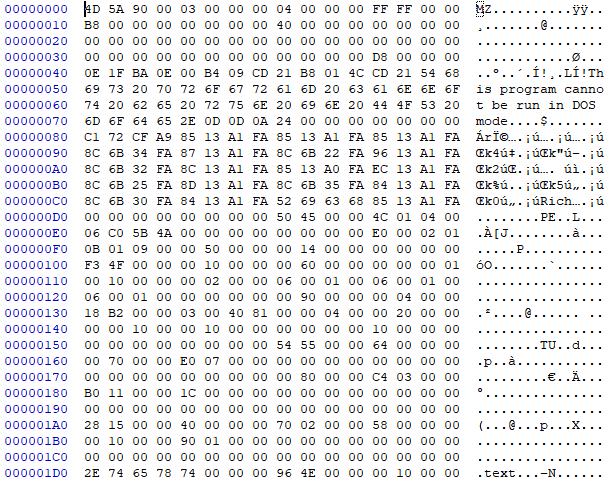
1) Extract various pe file format information as shown in Section 2 to 7 for calc.exe. What sections does it have? What are the addresses of the sections in the process image and in the physical file? What DLLs have been imported in calc.exe? What APIs are called for each DLL in calc.exe?

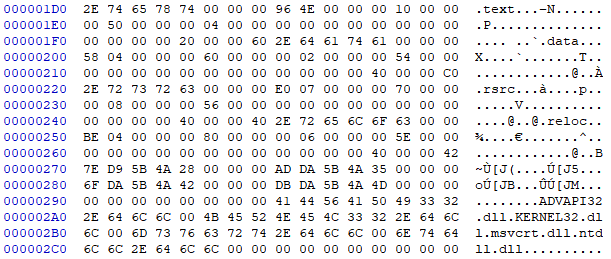
PE=dos header + image nt header + section table + sections



NT header(optional header)

dos header

NT header (sig + file header)



Section header table

(everything in little endian = you have to flip it to get real one)

image dos header

e\_magic : 4d 5a (0-1)

e\_cblp : 90 00 (2-3)

e\_lfanew : d8 00 00 00 (tells us the start of NT header) (3c-ef)

image nt header (sig + file header) (starts from d8 as e\_lfanew says)

pe signature : 50 45 00 00 (d8-db)

machine : 4c 01 (dc-dd)

number of sectors : 04 00 (0x4 = 4 sectors) (de-df)

size of optional header : e0 00 (0xe0 = 224bytes) (ec-ed)

image NT header (optional header) (d8 + 4 + 14 = f0)

magic : 0b 01 (f0-f1)

linker version : 09 00 (f2-f3)

address of entry point : f3 4f 00 00 (100-103)

number of entries in data directory : 10 00 00 00 (0x10 = 16) (14c-14f)

export directory rva : 00 00 00 00 (150-153)

export directory size : 00 00 00 00 (154-157)

import directory rva : 54 55 00 00 (158-15b)

import directory size : 64 00 00 00 (15c-15f)

Each section header=name(8)+misc(4)+ relative virtual address(4)+size(4)+file address(4)+ptr to reloc(4)+ptr to linenum(4)+num reloc(2)+num linenum(2)+characteristics(4)

Section header table (number of sectors = 4)

Section 1

name : 2e 74 65 78 74 00 00 00 (.text)

rva : 00 10 00 00 (0x0001000)

virtual address : 0x1001000 (0x1000000 + 0x0001000)

size : 00 50 00 00

file address : 00 04 00 00

Section 2

name : 2e 64 61 74 61 00 00 00 (.data)

rva : 00 60 00 00

virtual address : 0x1006000

size : 00 02 00 00

file address : 00 54 00 00

Section 3

name : 2e 72 73 72 63 00 00 00 (.rsrc)

rva : 00 70 00 00

virtual address : 0x1007000

size : 00 08 00 00

file address : 00 56 00 00

Section 4

name : 2e 72 65 6c 6f 63 00 00 (.reloc)

rva : 00 80 00 00

virtual address : 0x1008000

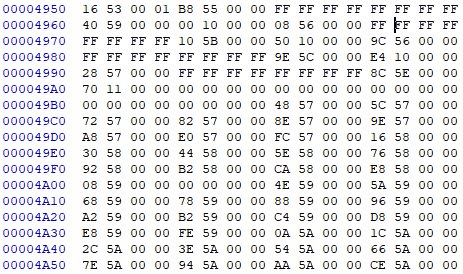
size : 00 06 00 00

file address : 00 5e 00 00

Since the rva of import directory is 0x5554, if we compute the file address from this rva, we can get 0x4954 (5554-1000+400) (file address) and that is the beginning of Image import descriptor which contains information about all of the dlls’.

(1000 is rva of owner section of x(import directory rva) which is .text, and 400 is the file address of owner section of x.)

(.text is the owner section because 0x1005554 is in between 0x1001000 and 0x1006000)



IMAGE\_IMPORT\_DESCRIPTOR(20 byte)=

OriginalFirstThunk(4)+TimeDateStamp(4)+ForwarderChain(4)+Name(4)+FirstThunk(4)

The first pointer of dll that points to the name is 0x5940 (name의 위치를 가리키는 rva) as we can find in the location 0x4960 (4954+4+4+4 = Name(4)). That is the rva and if we compute the file address from 0x5940, we can find out that it is 4D40.



The first dll that we can find is ADVAPI32.dll, and the rest of the dlls’ can be found in the way just like this.

Original First Thunk is the rva that points to the location that has the collection of APIs. For the first one, it is 0x55b8 and if we compute file address from this rva, it is 0x49b8.

(각 dll의 api들은 다 original first thunk를 보고 찾아냄)



There we can find 0x5748 at the location 0x49b8, which is again the rva, and if we compute the file address from 0x5748, which is 0x4b48, and go there, we can find 2 bytes of hint and the name of the first API. The next API is at the rva of 0x575c and we can go to find out the API name in the similar way.

DLL 1 (4D40)

Original First Thunk : B8 55 00 00

Time Date Stamp : FF FF FF FF

Forwarder Chain : FF FF FF FF

Name : ADVAPI32.dll (rva = 40 59 00 00)

First Thunk : 00 10 00 00

APIs’ : LookupAccountSidW, LookupAccountNameW, 6.GetLengthSid

DLL 2 (4F10)

Original First Thunk : 08 56 00 00

Time Date Stamp : FF FF FF FF

Forwarder Chain : FF FF FF FF

Name : KERNEL32.dll (rva = 10 5B 00 00)

First Thunk : 50 10 00 00

APIs’ : D.LocalAlloc, ..GetLastError, s.SetLastError

DLL 3 (509E)

Original First Thunk : 9C 56 00 00

Time Date Stamp : FF FF FF FF

Forwarder Chain : FF FF FF FF

Name : msvcrt.dll (rva = 9E 5C 00 00)

First Thunk : E4 10 00 00

APIs’ : Þ.malloc, ó.printf, Ð.\_vsnwprintf\_s

DLL 4 (528C)

Original First Thunk : 28 57 00 00

Time Date Stamp : FF FF FF FF

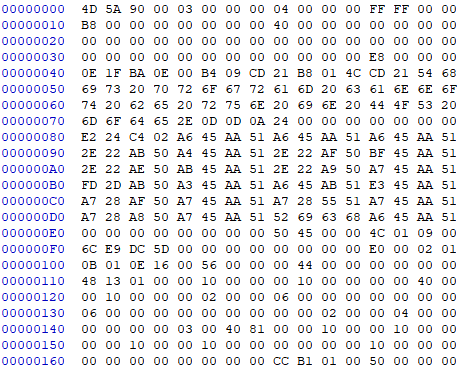
Forwarder Chain : FF FF FF FF

Name : ntdll.dll (rva = 8C 5E 00 00)

First Thunk : 70 11 00 00

APIs’ : S.RtlReleaseRelativeName, RtlNtStatusToDosError, Ô.NtClose.‹

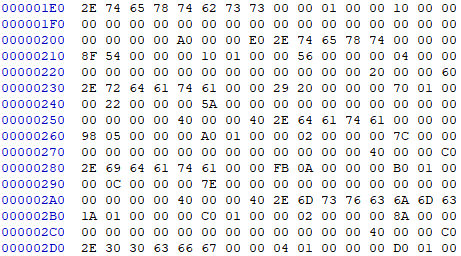
2) Compile displayAllModules in Section 10. Repeat 1) for displayAllModules.exe.



NT header(optional header)

NT header (sig + file header)

dos header



Section header table

image dos header

e\_magic : 4d 5a

e\_cblp : 90 00

e\_lfanew : e8 00 00 00 (start of NT header)

image nt header (sig + file header)

pe signature : 50 45 00 00

machine : 4c 01

number of sectors : 09 00 (0x9 = 9 sectors)

size of optional header : e0 00 (0xe0 = 224bytes)

image NT header (optional header) (e8 + 4 + 14)

magic : 0b 01

linker version : 0e 16

address of entry point : 48 13 01 00

number of entries in data directory : 10 00 00 00 (0x10 = 16)

export directory rva : 00 00 00 00

export directory size : 00 00 00 00

import directory rva : cc b1 01 00

import directory size : 50 00 00 00

Each section header=name(8)+misc(4)+relative virtual address(4)+size(4)+file address(4)+ptr to reloc(4)+ptr to linenum(4)+num reloc(2)+num linenum(2)+characteristics(4)

Section header table

Section 1

name : 2e 74 65 78 74 62 73 73 (.textbss)

rva : 00 10 00 00

virtual address : 0x1001000

size : 00 00 00 00

file address : 00 00 00 00

Section 2

name : 2e 74 65 78 74 00 00 00 (.text)

rva : 00 10 01 00

virtual address : 0x1011000

size : 00 56 00 00

file address : 00 04 00 00

Section 3

name : 2e 72 64 61 74 61 00 00 (.rdata)

rva : 00 70 01 00

virtual address : 0x1017000

size : 00 22 00 00

file address : 00 5a 00 00

Section 4

name : 2e 64 61 74 61 00 00 00 (.data)

rva : 00 a0 01 00

virtual address : 0x101a000

size : 00 02 00 00

file address : 00 7c 00 00

Section 5

name : 2e 69 64 61 74 61 00 00 (.idata)

rva : 00 b0 01 00

virtual address : 0x101b000

size : 00 0c 00 00

file address : 00 7e 00 00

Section 6

name : 2e 6d 73 76 63 6a 6d 63 (.msvcjmc)

rva : 00 c0 01 00

virtual address : 0x101c000

size : 00 02 00 00

file address : 00 8a 00 00

Section 7

name : 2e 30 30 63 66 67 00 00 (.00cfg)

rva : 00 d0 01 00

virtual address : 0x101d000

size : 00 02 00 00

file address : 00 8c 00 00

Section 8

name : 2e 72 73 72 63 00 00 00 (.rsrc)

rva : 00 e0 01 00

virtual address : 0x101e000

size : 00 06 00 00

file address : 00 8e 00 00

Section 9

name : 2e 72 65 6c 6f 63 00 00 (.reloc)

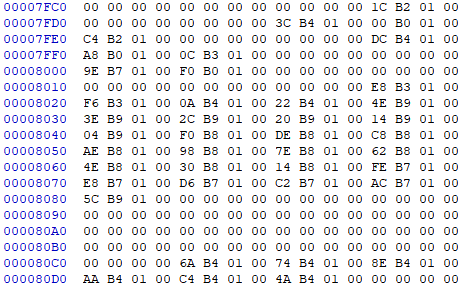
rva : 00 f0 01 00

virtual address : 0x101f000

size : 00 06 00 00

file address : 00 94 00 00

The virtual address of import directory is 0x101b1cc, which is between virtual address of .idata and .msvcjmc. Therefore, we can conclude that the owner section of import directory is .idata. The rva of the owner section is 0x1b000, and file address of the owner section is 0x7e00. So the file address of import directory, which is the beginning of image import descriptor, is 1b1cc-1b000+7e00 = 0x7fcc.



IMAGE\_IMPORT\_DESCRIPTOR(20 byte)=

OriginalFirstThunk(4)+TimeDateStamp(4)+ForwarderChain(4)+Name(4)+FirstThunk(4)

At the location of 0x7fd8 (7fcc+4+4+4), we can find out that the name pointer of the first dll points to 0x1b43c, and the file address of it is 0x823c, which is location of the first dll.

DLL 1 (823C)

Original First Thunk : 1c b2 01 00

Time Date Stamp : 00 00 00 00

Forwarder Chain : 00 00 00 00

Name : KERNEL32.dll (rva = 3c b4 01 00)

First Thunk : 00 b0 01 00

APIs’ : t.CloseHandle, ..GetCurrentProcess, —.K32EnumProcessModules

DLL 2 (82DC)

Original First Thunk : c4 b2 01 00

Time Date Stamp : 00 00 00 00

Forwarder Chain : 00 00 00 00

Name : VCRUNTIME140D.dll (rva = 10 5B 00 00)

First Thunk : a8 b0 01 00

APIs’ : H.memset, 5.\_except\_handler4\_common, ..\_\_vcrt\_GetModuleFileNameW

DLL 3 (859E)

Original First Thunk : 0c b3 01 00

Time Date Stamp : 00 00 00 00

Forwarder Chain : 00 00 00 00

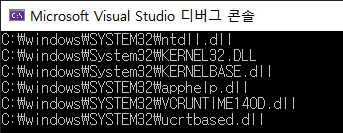
Name : ucrtbased.dll (rva = 9e b7 01 00)

First Thunk : f0 b0 01 00

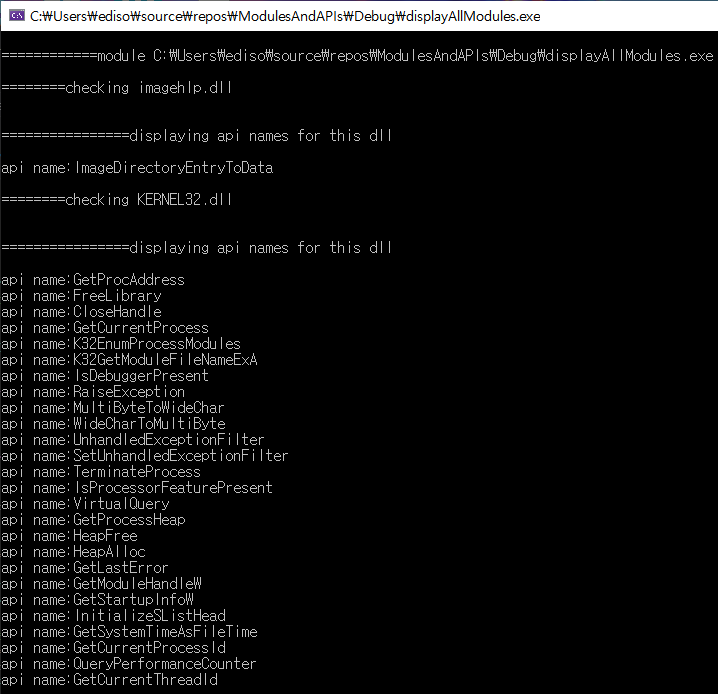
APIs’ : s.\_\_p\_\_commode, I.strcpy\_s, \_exit

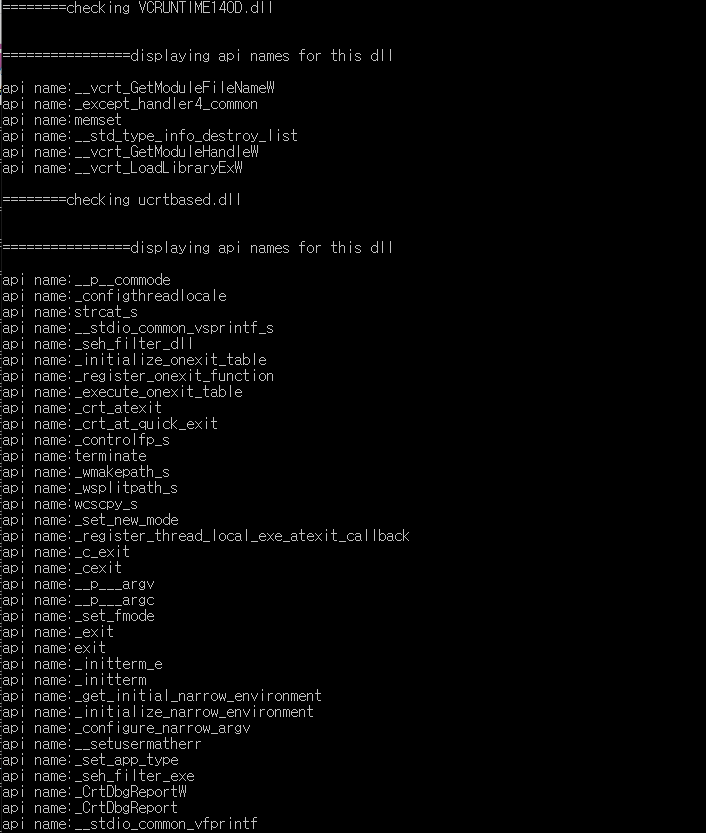
3) Run the program in Sect 10 and 11 and compare the results with those in Problem 2).

Program in section 10



Program in section 11





이 외에도 수없이 많은 api name들이 출력됐다.

2번 문제와 비교해봤을 때, 2번에서는 총 3개의 dll밖에 출력되지 않았지만 3번에서는 6개의 dll이 출력된걸 볼 수 있다. 3번에서 더 많이 출력되는 이유는 단순 분석이 아닌 프로그램을 실제로 돌려보면 dll이 또다른 dll을 부르기 때문이고, 분석이 아닌 실제로 프로그램을 돌려봤을 때 모든 dll이 다 출력된다고 볼 수 있다. API 역시 2번보다 3번에서 훨씬 더 많이 출력된걸 볼 수 있다.