

IDA Pro Fundamental

강사. 유효곤
(ugonfor@gmail.com)

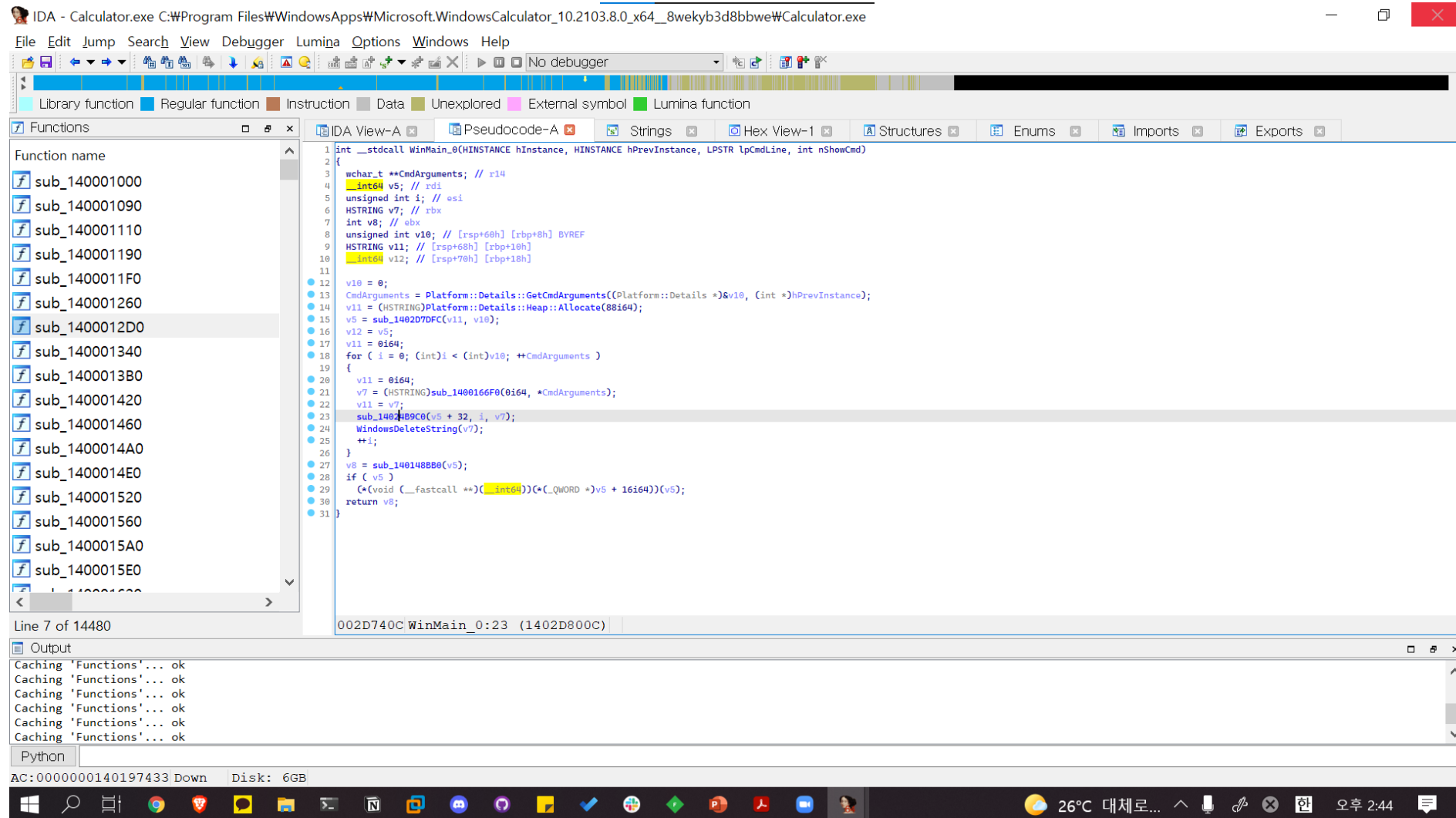
Contents

- IDA Pro란,
- IDA 기본 사용법
- IDA 단축키
- IDA Debugger
- IDA Python
- IDA 외 리버싱 툴들

IDA Pro란,

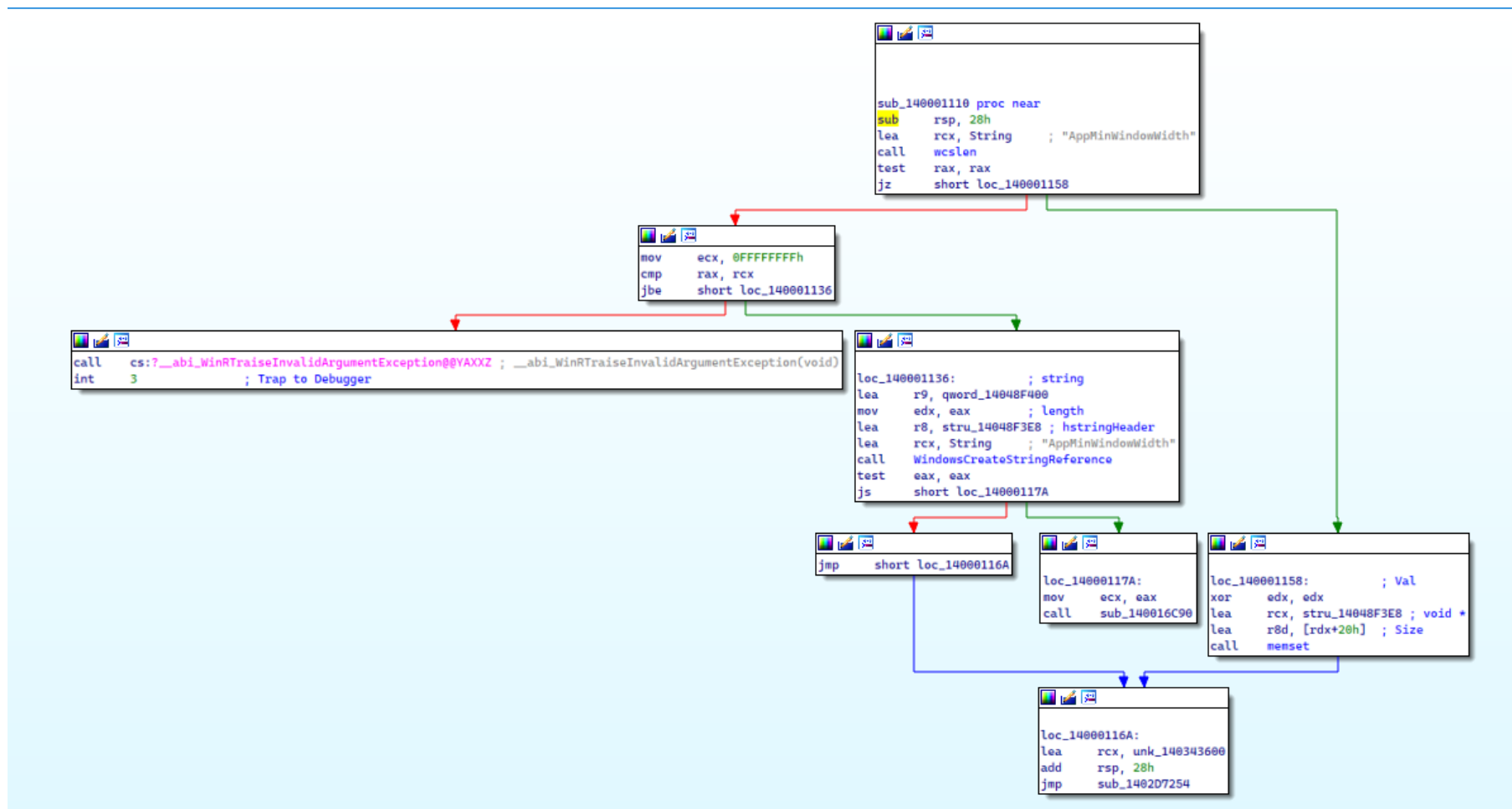
IDA Pro란,

디스어셈블러 및 디컴파일러



IDA Pro란,

디스어셈블러 및 디컴파일러



IDA Pro란,

디스어셈블러 및 디컴파일러

```
.text:0000000140001110 ; ===== S U B R O U T I N E =====
.text:0000000140001110
.text:0000000140001110
.text:0000000140001110 sub_140001110 proc near ; DATA XREF: .pdata:0000000140492018↓o
.text:0000000140001110 sub     rsp, 28h
.text:0000000140001114     lea     rcx, String ; "AppMinWindowWidth"
.text:0000000140001118     call    wcslen
.text:0000000140001120     test    rax, rax
.text:0000000140001123     jz      short loc_140001158
.text:0000000140001125     mov     ecx, 0FFFFFFFFh
.text:000000014000112A     cmp     rax, rcx
.text:000000014000112D     jbe     short loc_140001136
.text:000000014000112F     call    cs:??_abi_WinRTraiseInvalidArgumentException@@YAXXZ ; __abi_WinRTraiseInvalidArgumentException(void)
.text:0000000140001135     int     3 ; Trap to Debugger
.text:0000000140001136 ; -----
.text:0000000140001136 loc_140001136: ; CODE XREF: sub_140001110+1D↑j
.text:0000000140001136     lea     r9, qword_14048F400 ; string
.text:000000014000113D     mov     edx, eax ; length
.text:000000014000113F     lea     r8, stru_14048F3E8 ; hstringHeader
.text:0000000140001146     lea     rcx, String ; "AppMinWindowWidth"
.text:000000014000114D     call    WindowsCreateStringReference
.text:0000000140001152     test    eax, eax
.text:0000000140001154     js      short loc_14000117A
.text:0000000140001156     jmp     short loc_14000116A
.text:0000000140001158 ; -----
.text:0000000140001158 loc_140001158: ; CODE XREF: sub_140001110+13↑j
.text:0000000140001158     xor     edx, edx ; Val
.text:000000014000115A     lea     rcx, stru_14048F3E8 ; void *
.text:0000000140001161     lea     r8d, [rdx+20h] ; Size
.text:0000000140001165     call    memset
.text:000000014000116A
.text:000000014000116A loc_14000116A: ; CODE XREF: sub_140001110+46↑j
.text:000000014000116A     lea     rcx, sub_140343600 ; void (__cdecl *)(void)
.text:0000000140001171     add     rsp, 28h
.text:0000000140001175     jmp     atexit
.text:000000014000117A ; -----
```

IDA Pro란,

디스어셈블러 및 디컴파일러

IDA Pro는 결국,

- 바이너리를 사람이 볼 수 있는 형태로 바꿔주는 것!
 - Machine code -> Human Readable !
 - Disassembler, Decompiler

IDA Pro란,

컴파일러란,

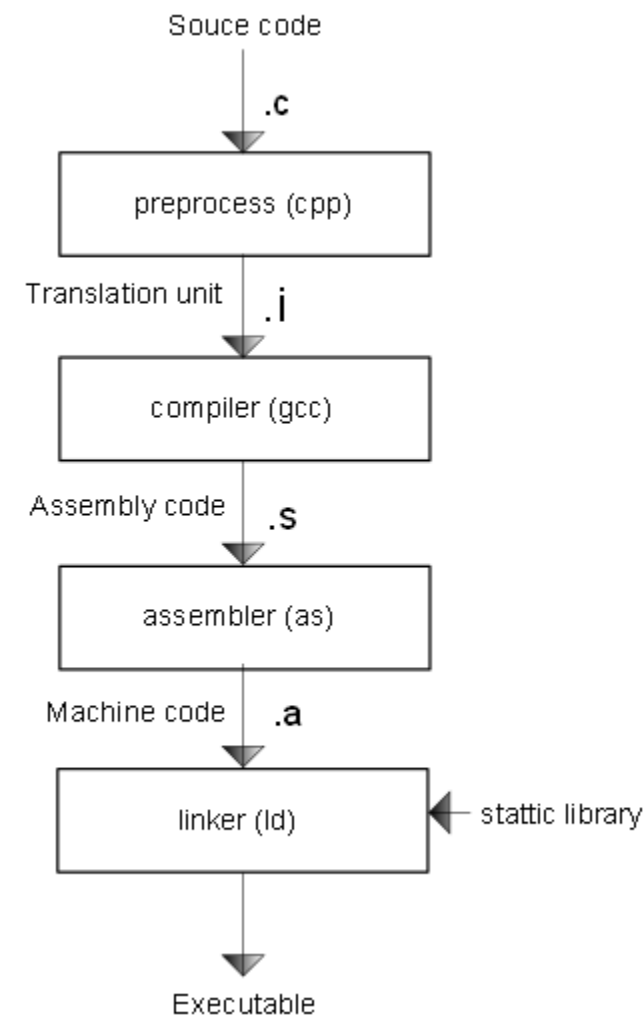
\$ gcc -o main main.c

\$ gcc -c main.c func.c

\$ gcc -o main main.o func.o

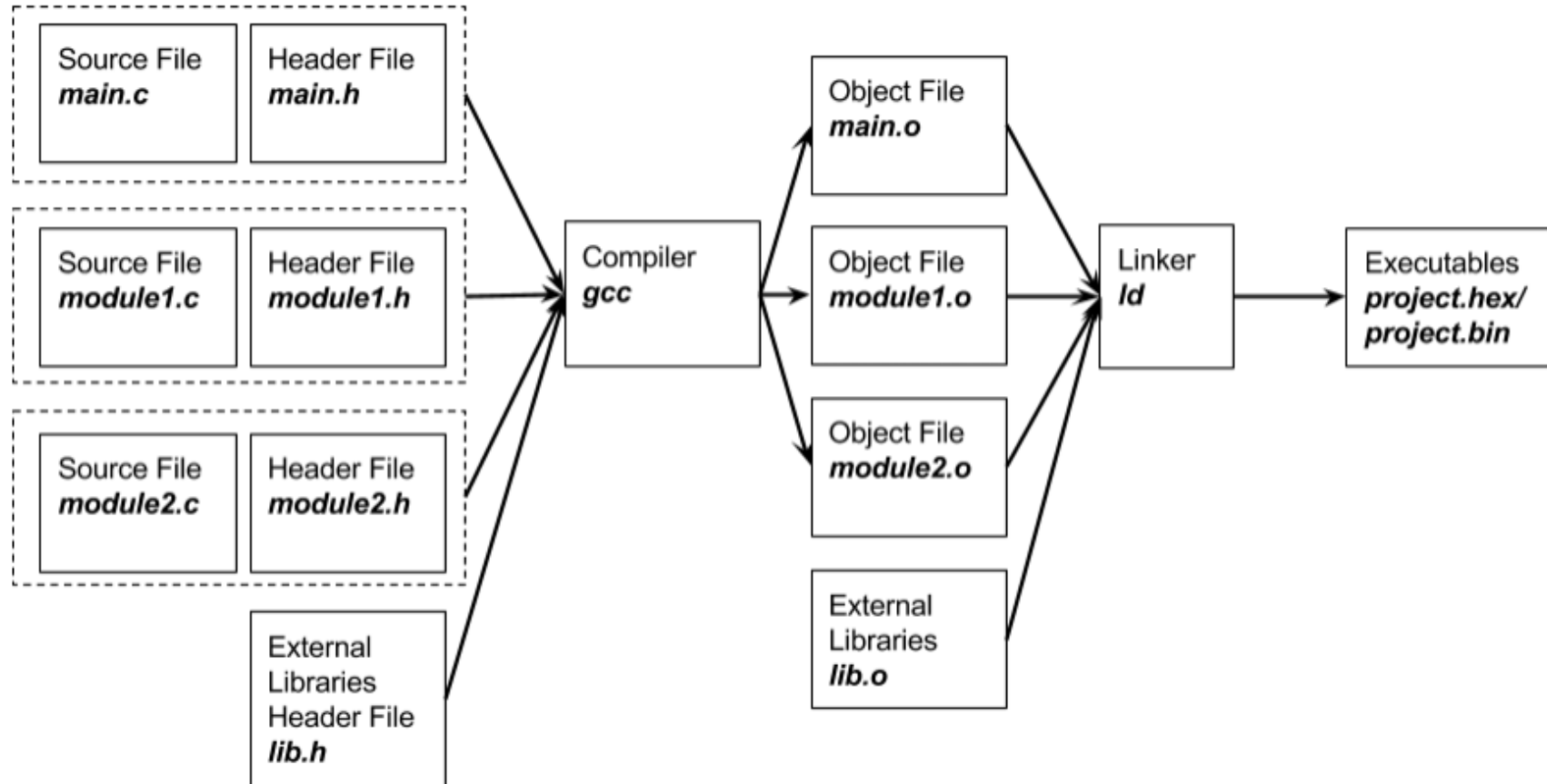
Source code → (preprocessing) → Assembly → Object file → Binary

main.c → (main.i) → main.s → main.o → main(.bin)



IDA Pro란,

컴파일러란,



IDA Pro란,

컴파일러란,

Source code → (preprocessing) → Assembly → Object file → Binary

main.c → (main.i) → main.s → main.o → main(.bin)

어떤 컴퓨터 프로그램을 분석하기 위해서는 어셈블리 / 소스코드 수준의 코드가 필요함.

이보다 더 저수준 언어의 경우에는 사람이 읽을 수 없음.

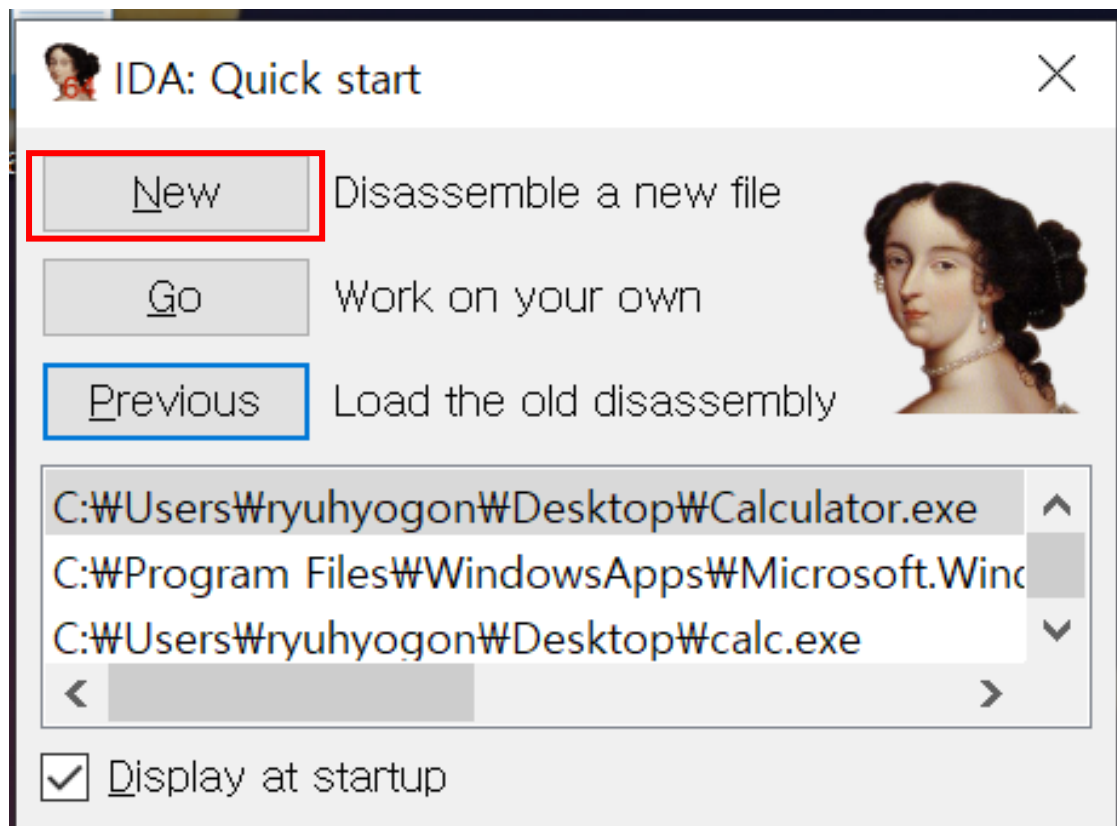
그래서 저수준 코드를 사람이 읽을 수 있게 만들어주는 것: **disassembler, decompiler**

Disassembly 보다 decompile을 잘하는 것이 **훨씬** 고난이도의 작업 : The reason why IDA is the best tool.

IDA 기본 사용법

IDA 기본 사용법

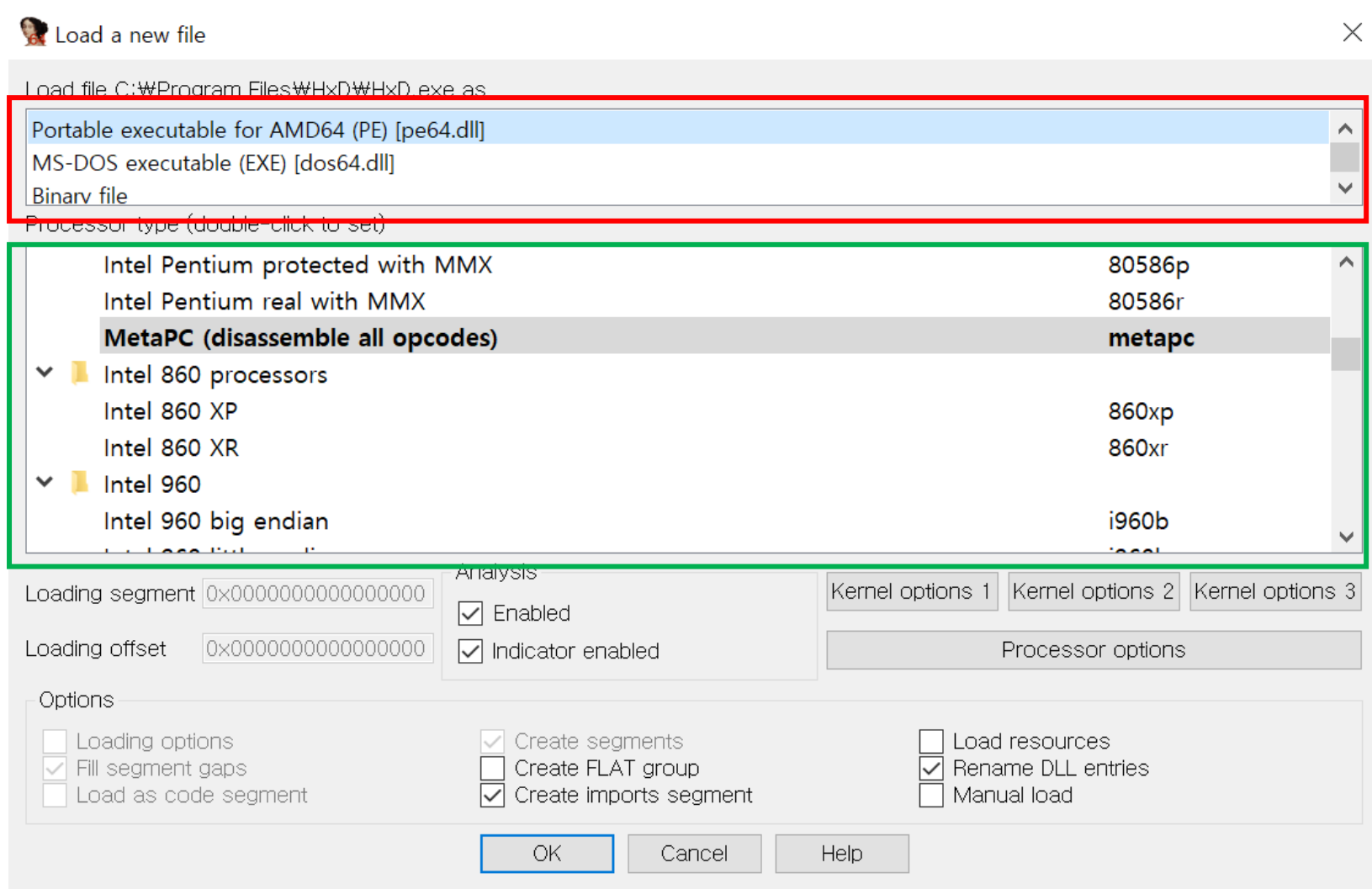
파일 로드



- 분석하고자 하는 파일 선택

IDA 기본 사용법

파일 로드



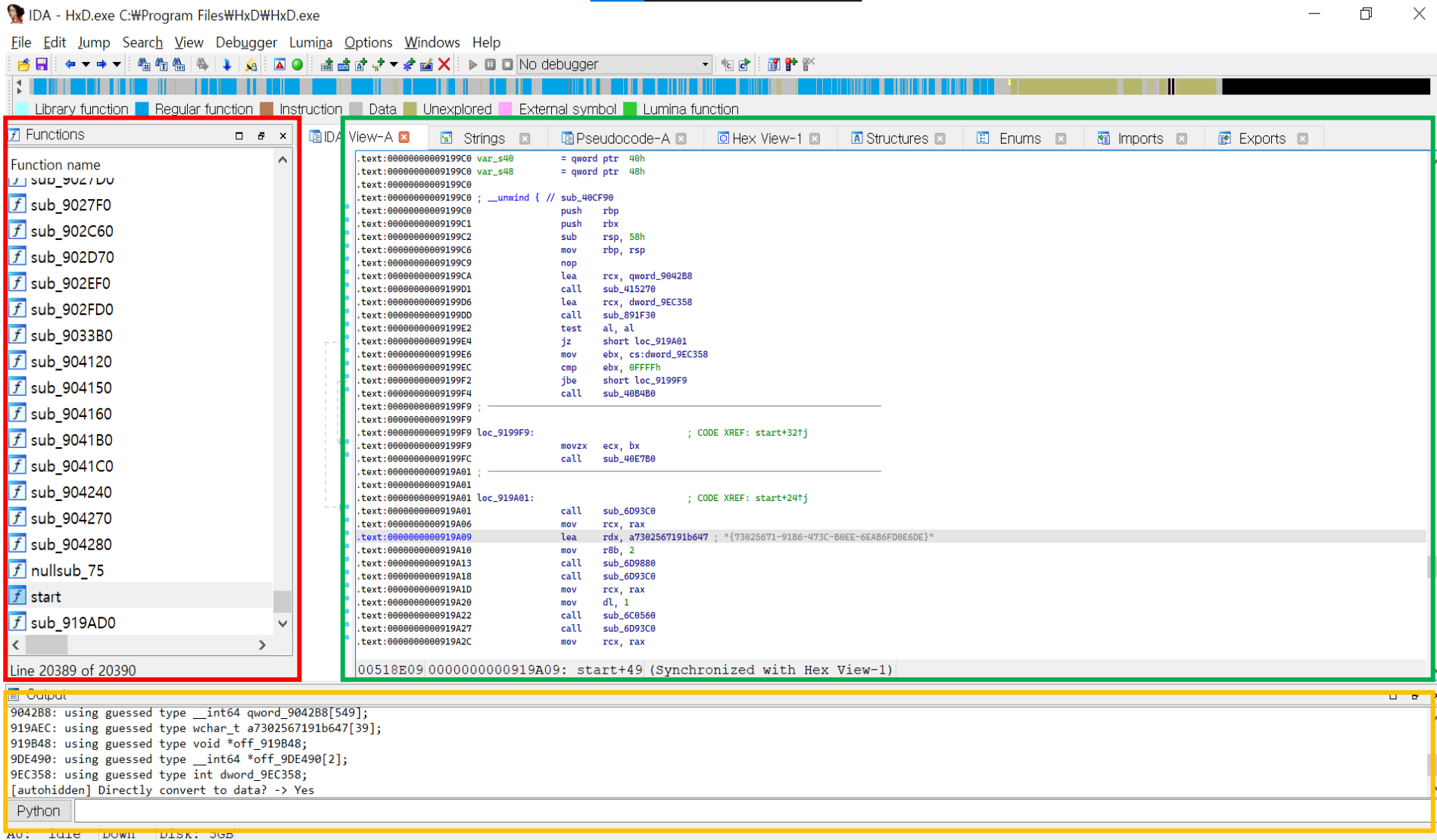
□: 파일의 종류

□: CPU 프로세서 종류

→ 거의 만질 일 없음.

IDA 기본 사용법

파일 로드



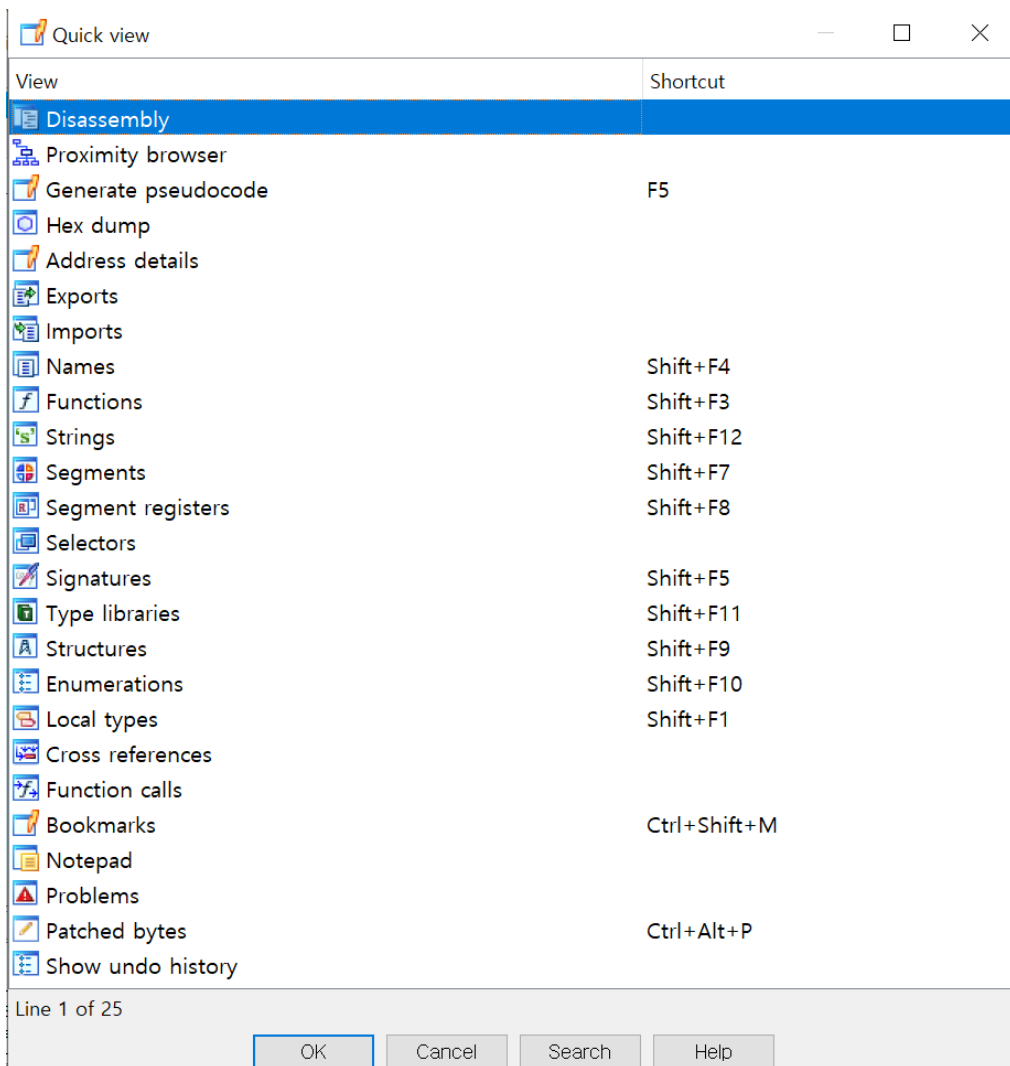
□: 함수 목록

□: 메인 화면

□: output view

IDA 기본 사용법

메인 기능

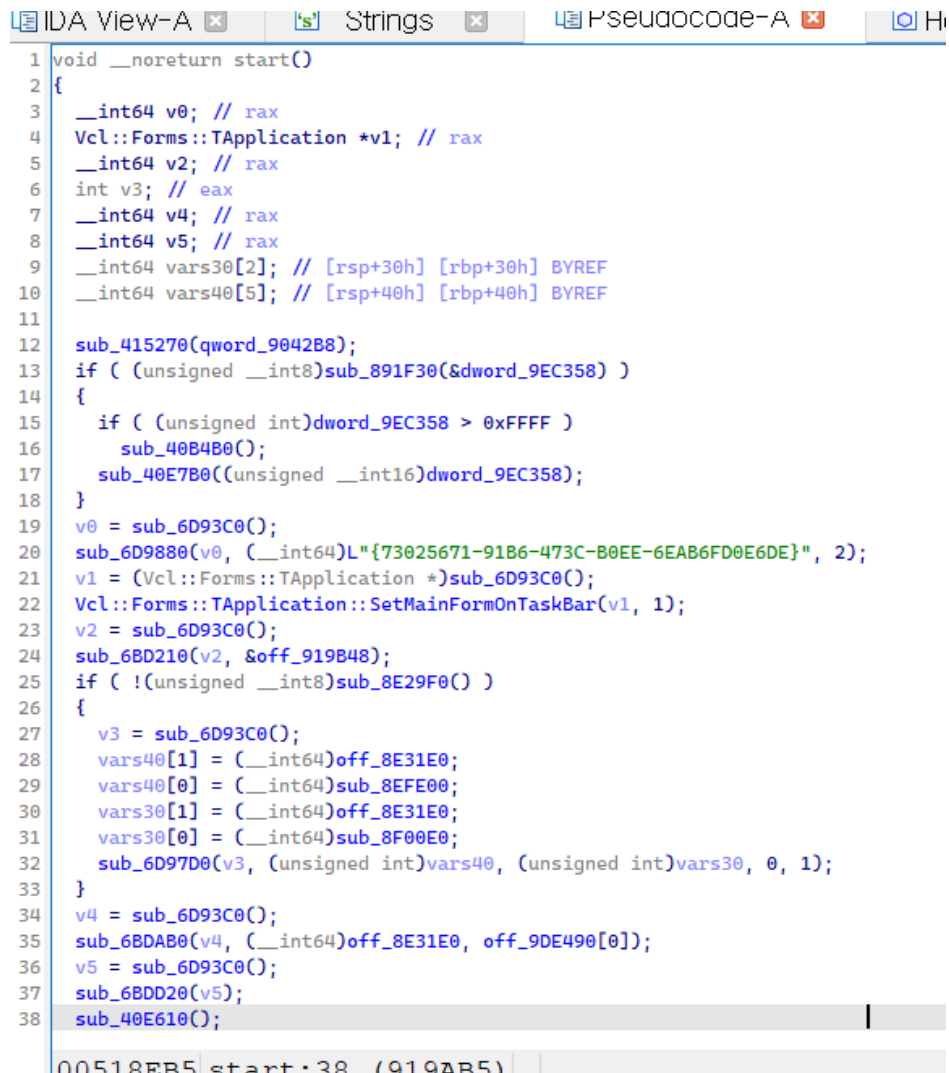


자주 사용되는 기능

- Disassembly : 디스어셈블리(어셈블리) 코드
- Generate pseudocode : 디컴파일(수도코드) 코드
- Hex dump : Hex View
- Exports : Export 함수들 목록
- Imports : Import 함수들 목록
- Functions : 바이너리 내 정의된 함수 목록
- Strings : 바이너리 내 문자열 목록 (strings와 동일 기능)
- Structures : 구조체 선언
- Graph view : 어셈블리 코드를 그래프 형태로 보여줌 (CFG)

IDA 기본 사용법

메인 기능



```
1 void __noreturn start()
2 {
3     __int64 v0; // rax
4     Vcl::Forms::TApplication *v1; // rax
5     __int64 v2; // rax
6     int v3; // eax
7     __int64 v4; // rax
8     __int64 v5; // rax
9     __int64 vars30[2]; // [rsp+30h] [rbp+30h] BYREF
10    __int64 vars40[5]; // [rsp+40h] [rbp+40h] BYREF
11
12    sub_415270(qword_9042B8);
13    if ( (unsigned __int8)sub_891F30(&dword_9EC358) )
14    {
15        if ( (unsigned int)dword_9EC358 > 0xFFFF )
16            sub_40B4B0();
17        sub_40E7B0((unsigned __int16)dword_9EC358);
18    }
19    v0 = sub_6D93C0();
20    sub_6D9880(v0, (__int64)L"{73025671-91B6-473C-B0EE-6EAB6FD0E6DE}", 2);
21    v1 = (Vcl::Forms::TApplication *)sub_6D93C0();
22    Vcl::Forms::TApplication::SetMainFormOnTaskBar(v1, 1);
23    v2 = sub_6D93C0();
24    sub_6BD210(v2, &off_919B48);
25    if ( !(unsigned __int8)sub_8E29F0() )
26    {
27        v3 = sub_6D93C0();
28        vars40[1] = (__int64)off_8E31E0;
29        vars40[0] = (__int64)sub_8EFE00;
30        vars30[1] = (__int64)off_8E31E0;
31        vars30[0] = (__int64)sub_8F00E0;
32        sub_6D97D0(v3, (unsigned int)vars40, (unsigned int)vars30, 0, 1);
33    }
34    v4 = sub_6D93C0();
35    sub_6BDAB0(v4, (__int64)off_8E31E0, off_9DE490[0]);
36    v5 = sub_6D93C0();
37    sub_6BDD20(v5);
38    sub_40E610();
```

Pseudocode를 보여줌

- 함수 호출, 변수 등을 자동으로 분석하여 보여줌.
- 각 함수에 대해서 명칭 및 인자로 들어가는 값들의 자료형 들도 정의 가능하며, 이에 따라 자동으로 분석
- 구조체 선언도 가능
- 주석도 달 수 있음

IDA 단축키

IDA 단축키

반드시 알아야 하는 단축키

- F5 : 함수 디컴파일 (기본적으로 아이디가 깔려 있는 사람이라면 사용할 줄 아는 것)
- Tab : 수도코드 <-> IDA View 전환 (이때 커서가 가리키고 있던 부분이 그대로 유지됨!)
- n : 변수명, 함수명 변경
- Space : 그래프 표시
- ESC : 이전 화면으로 돌아가기
- Shift + F12 : String window 보여줌
- h : 숫자 Dec <-> Hex 변환
- R : 숫자 <-> Ascii 변환 (0x61 <-> 'a' 이런식으로 변환됨)
- / : 수도코드에 주석 달기
- ; : 어셈블리에 주석달기
- G : 커서를 원하는 주소로 보내줌

IDA 단축키

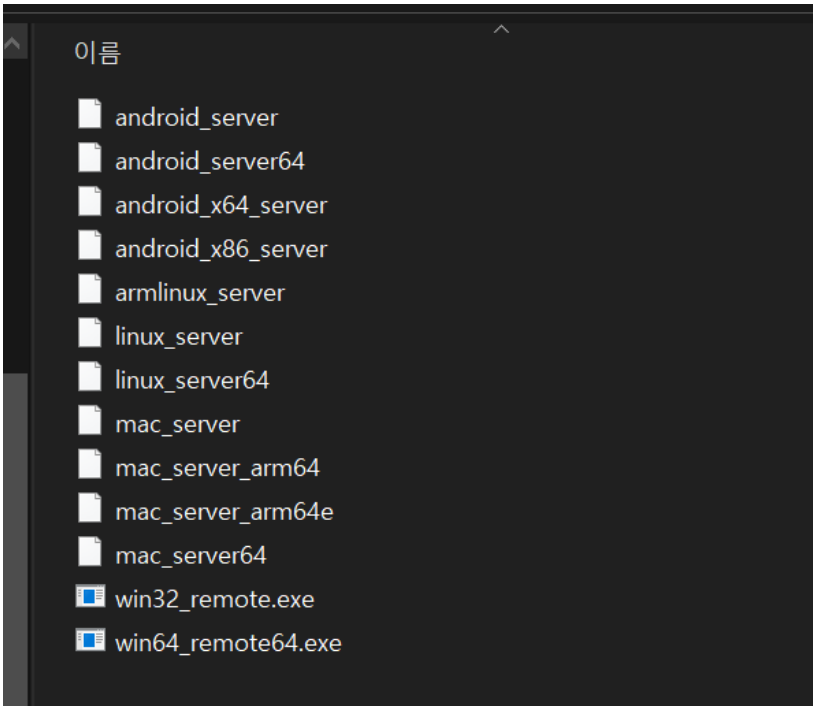
반드시 알아야 하는 단축키

- X: Xref 표시 (이거 진짜 개 중요함. 제일 많이 쓰는 기능임. 변수나 함수 아무거나 놓고 눌러보셈)
- Ctrl + X: Xref 표시 (조금 달랐던거 같긴함)
- D: 자료형 변환 (데이터에서 Byte - Word - Dword 변환해줌)
- *: 배열 선언
- C: 데이터를 코드로 변환
- P: 함수로 지정 (함수 에필로그가 망가져 있으면 안 되는 경우가 많음)
- E: 함수 종료 지점 변경
- A: 데이터를 문자열로 변경
- U: undefine (데이터를 그냥 데이터로 인식. 코드나 문자열, 배열로 선언 된 것들을 초기화 시킴)
- F2: 데이터 패치 (보통, Hex View에서 하는 걸 추천함)
- Ctrl + 1: 창 선택하게 해 줌.
- Y: 변수의 자료형 변환
- Ctrl: 스크롤 빨라짐

IDA Debugger

IDA Debugger

환경설정

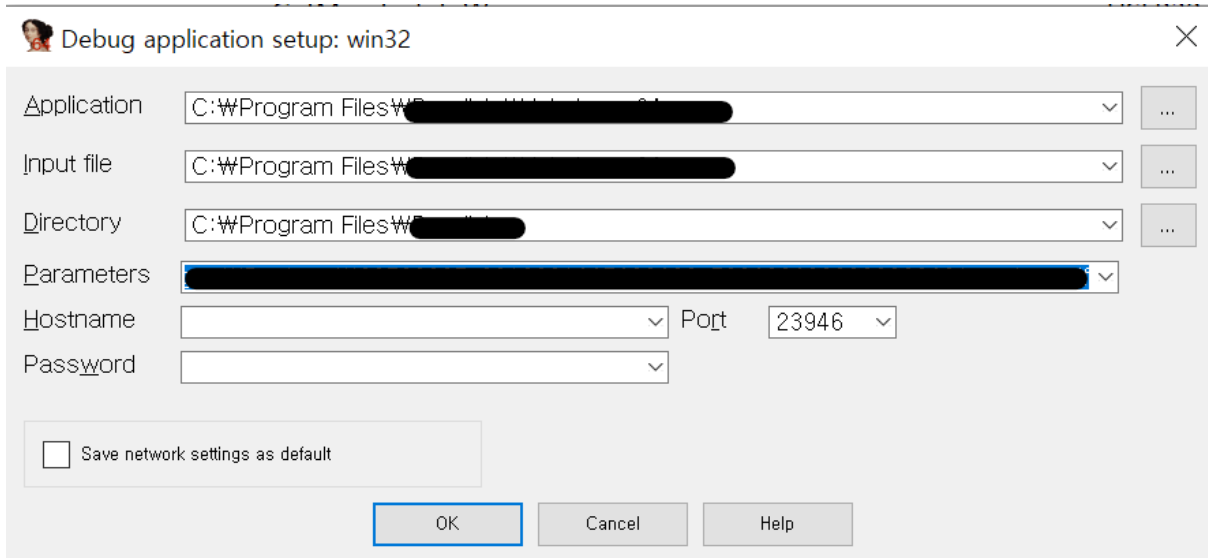
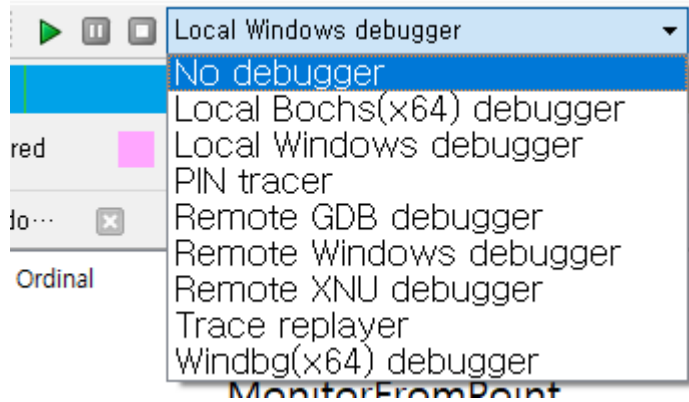


- /\$아이다가_설치된_경로/dbgsrv/ 에 있는 서버 프로세스
- 사실 아이피를 확인 한 후 프로세스 실행

```
ugonfor@gongon:/windir/c/Program Files/IDA Pro 7.6/dbgsrv$ ./linux_server
IDA Linux 32-bit remote debug server(ST) v7.6.27. Hex-Rays (c) 2004-2021
Listening on 0.0.0.0:23946...
```

IDA Debugger

환경설정



- 디버거를 선택한 후,
- Debugger 탭 - Option에서 왼쪽 화면 과 같이 Application, Input File, Directory, Parameter를 적절히 지정해 준 후,
- Hostname에 사설 아이피를 입력
- 이후 디버깅 가능

만약, 로컬환경에서 디버깅할 것이면, Local Windows Debugger 선택하면 됨.

IDA Debugger

■ 디버깅 단축키

- F9 : Start Process
- F8 : Step Over
- F7 : Step into
- F2 : Breakpoint
- Ctrl + F7 : Run until Return
- Ctrl + 2 : Debugger관련 창 목록
- Ctrl + Alt + s : 콜스택 확인

IDA Python

IDA Python

■ IDAPython

- 그러면, IDA를 사용하면 뭐든지 다 가능한가?
- 가능하다! 물론 “시간이 무한하면”
- 만약, 동일 작업을 1000번 정도 반복해야 하면 어떻게 하는 가?
- 자동화가 필요함.
- 이를 해주는 것이 IDAPython.
- IDA의 경우, 대부분의 작업을 Python API로 만들어 놓았기에, 하고자 하는 것을 파이썬 스크립트로 만들 수 있음.

IDA Python

Cheat Sheet

Some constants

```
BADADDR = BADSEL = 0xffffffffL
MAXADDR = 0xFFFFFFFF
SIZE_MAX = 0xffffffffL
```

Analysis

```
plan_and_wait(start, end) AnalyzeRange
auto_mark_range(start, end, QType)
delete_all_segments()
demangle_name(name, disableMask)
```

Entry points

```
*Entries()
add_entry(ord, ea, name, makeCode)
rename_entry(ord, name)
get_entry(ord)
get_entry_ordinal(index)
get_entry_qty()
```

Cross references (XRef)

```
= Code refs = *CodeRefsTo(ea, flow)
to / from addr
*CodeRefsFrom(ea, flow)

= Data refs = *DataRefsTo(ea)
to / from addr
*DataRefsFrom(ea)

= All refs = *XrefsTo(ea, flags=0)
to / from addr
*XrefsFrom(ea, flags=0)
```

Functions

```
*Functions(ea, end)
iterate
= over =
functions get_prev_func(ea)
get_next_func(ea)

get_name_ea_simple(name) Line address of name
get_func_name(ea) Func. name or empty string
get_func_off_str(ea) 'funcname+offset' string

= Manage function =
set_func_end(ea, end)
add_func(ea, end=BADADDR) del_func(ea)

= Function comment =
get_func_cmt(ea, rpt)
set_func_cmt(ea, cmt, rpt)

= Manage function attributes =
get_func_attr(ea, attr)
set_func_attr(ea, attr, val)

= Stack pointer interaction =
get_sp_delta(ea)
add_user_stkpnt(ea, delta)

= Function frame interaction =
set_frame_size(ea, lvsz, frregs, args)
get_frame_size(ea)
get_frame_regs_size(ea)
get_frame_lvar_size(ea)
get_frame_args_size(ea)
get_frame_id(ea) ID of function frame structure
```

User interface

```
= Get / set current position =
get_screen_ea() here() jump_to(ea)

= GUI-dialogs =
msg(msg) warning(msg)
ask_yn(dflt, prompt) choose_func(title)
idaapi.ask_str(dflt, history, prompt)
idaapi.ask_file(doSave, mask, prompt)
```

Search in database

```
find_binary(ea, flag, bin, rdx=16)
find_text(ea, flag, row, col, text)
find_imm(ea, flag, value)
find_data(ea, flag)
find_func_end(ea)
```

Items

```
next_addr(ea) next_prev_not_tail(ea)
get_item(ea) item* start/end addr
get_item_size(ea) from ea to the end
next_head(ea, maxea=BADADDR)
prev_head(ea, min=0)
*FuncItems(ea) Function items' addr
*Heads(ea=None, end=None) Items' addr from ea to end
```

Segments

```
get_segm_attr(ea, attr) get_segm_start(ea)
set_segm_attr(ea, attr, value)
get_segm_name(ea) selector_by_name(name)
= iterate over segments =
*Segments()
get_first_seg() get_next_seg(ea)
```

Read / Write in database

```
get_wide_word(ea) patch_word(ea, val)
get_dword(ea) patch_dword(ea, val)
get_qword(ea) patch_qword(ea, val)
get_bytes(ea, size, useDbg=False)
is_bytes(ea) is byte initialized?
```

Colors (background)

```
get_color(ea, what)
set_color(ea, what, color)

CIC.*
*ITEM (1) RGB
*FUNC (2) hex
*SEG (3) 0xBBGGRR
```

Debugger Hooks

```
add_bpt(ea, size=0, type=BPT_DEFAULT) del_bpt(ea)
enable_bpt(ea, flag)
get_bpt_qty() get_bpt_ea(n)
get_reg_value(reg) read_dbg_memory(ea, size)
set_reg_value(val, reg) write_dbg_memory(ea, data)
step_into() run_to(ea) read_dbg_word(ea)
step_over() read_dbg_dword(ea)
step_until_ret() read_dbg_qword(ea)
start_process(path, args, sdtr)
```

Enums

```
get_enum_qty() getn_enum(idx)
get_enum_size(id)
get_enum_member_by_name(name)
get_enum_member_value(id)
get_enum_name(id)
get_enum_member_name(id)
get_enum_member_cmt(id, rpt)
add_struct_member(id, name, mOff, flag, type, nbytes, tgt=-1, tdelta=0, rtype=REF_OFF32)
del_struct_member(id, mOff)
```

Structures

```
= Structure =
*Structs() add_struct(idx, name, isUnion)
get_struct_qty() del_struct(id) is_union(id)
get_struct_by_idx(idx)
get_struct_id(name) get_struct_name(id)
get_struct_size(id) get_struct_cmt(id, rpt)

= Structure fields =
get_member_qty(id)
get_member_strid(id, mOff)
get_member_name(id, mOff)
get_member_cmt(id, mOff, rpt)
set_member_name(id, mOff, name)
set_member_cmt(id, mOff, cmt, rpt)
```

Listing, comments, operands

```
= Comments =
set_cmt(ea, cmt, rpt)
get_cmt(ea, rpt)

= Listing interaction =
generate_disasm_line(ea, flags)
print_insn_mnem(ea)
print_operand(ea, n)
get_operand_value(ea, n)
get_operand_type(ea, n)
set_manual_insn(ea, insn)
get_manual_insn(ea)

= Operand interpretation =
toggle_sign(ea, n) Change sign / bitwise not
op_stroff(ea, n, stid, delta)
op_enum(ea, n, enmid, serial)
op_plain_offset(ea, n, base)
op_offset(ea, n, rtype, tgt, base, tdelta)

= Name of code / data =
*Names()
set_name(ea, name, sn_flags=SN_CHECK)
get_name(ea, gtn_flags=0)

= Strings =
*Strings()
get_str_type(ea)
get_strlit_contents(ea, len=1, stype=0)
ida_bytes.create_strlit(ea, len, stype)
```

IDA Python 7.x

long char void bool iterator
More info in modules at IDA_DIR\python\Backward compatibility: IDA_DIR\python\ida_b695.py

Python 2.7

infocion

Pavel Rusanov

- <https://github.com/inforion/idadpython-cheatsheet>

IDA Python

example1

```
38  memset(v4, 0, 0x104u);
39  memset(v15, 0, 0x104u);
40  memset(ProcName, 0, 0x104u);
41  memset(v13, 0, 0x104u);
42  sub_10001DC0("yvpqsr2hpp", (unsigned int)v15); // urlmon.dll
43  LibraryA = LoadLibraryA(v15);
44  sub_10001DC0("MrxivrixStirE", (unsigned int)v10); // InternetOpenA
45  sub_10001DC0("MrxivrixGpswilerhpi", (unsigned int)v9); // InternetCloseHandle
46  sub_10001DC0("MrxivrixGsrrigxE", (unsigned int)v7); // InternetConnectA
47  sub_10001DC0("LxxtStirViuyiwxE", (unsigned int)v8); // HttpOpenRequestA
48  sub_10001DC0("LxxtWirhViuyiwxE", (unsigned int)v11); // HttpSendRequestA
49  sub_10001DC0("YVPHs{rpsehXsJmpiE", (unsigned int)ProcName); // URLDownloadToFileA
50  sub_10001DC0("HipixiYvpGegliIrxv}E", (unsigned int)v13); // DeleteUrlCacheEntryA
51  dword_1001D238 = (int)GetProcAddress(LibraryA, ProcName);
52  dword_1001D234 = (int)GetProcAddress(v1, v13);
53  sub_10001DC0("Oivrip762hpp", (unsigned int)v12); // Kernel32.dll
54  sub_10001DC0("GviexiXssplipt76Wretwlsx", (unsigned int)v6); // CreateToolhelp32Snapshot
55  sub_10001DC0("Tvsgiww76Jmvwx", (unsigned int)v5); // Process32First
56  sub_10001DC0("Tvsgiww76Ri|x", (unsigned int)v4); // Process32Next
57  v3 = LoadLibraryA(v12);
```

- 문자열 난독화를 푸는 과정
 - 실행시켜보고 직접 확인하는 것도 한 가지 방법
 - 하지만, 악성코드기에 실행하기가 켄 어려울 수 있음.
 - IDAPython을 통해 해결
- 01_hwp101pe.py 참고

IDA Python

■ Example2. MidnightSun CTF 2021 Qual - labyrevnt

- 실습 진행

IDA외 리버싱 툴들

IDA 외 리버싱 툴들

IDA는 디스어셈블러!

- 리버싱을 하다보면, 수많은 툴들이 필요하게 됨
- 다양한 툴들을 사용할 줄 아는 것도 필수적이다.
 - PE 헤더 파서
 - 후킹
 - dotnet 디컴파일러
 - 메모리 덤프
 - 프로세스 모니터
 - APK 분석 툴
 - IDA Pro
 - SysinternalsSuite tools
 - Frida
 - HxD
 - Exeinfo
 - CheatEngine
 - Dnspy
 - X64dbg
 - ...

Q&A
