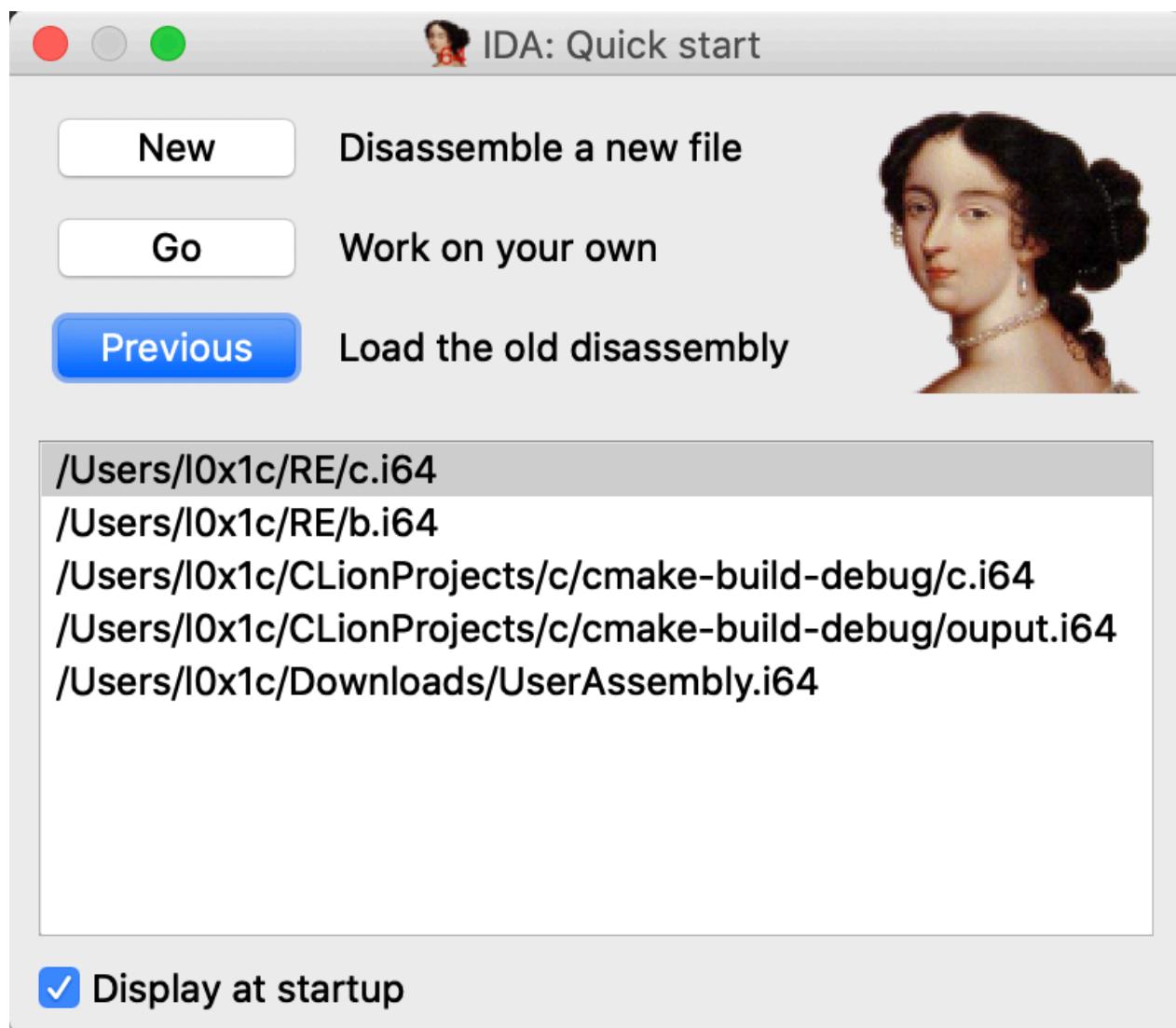


IDA Pro

网址：<https://www.youtube.com/watch?v=GEtpznMFsEs&list=PLKwUZp9HwWoDDBPvoapdbJ1rdofowT67z&index=3>

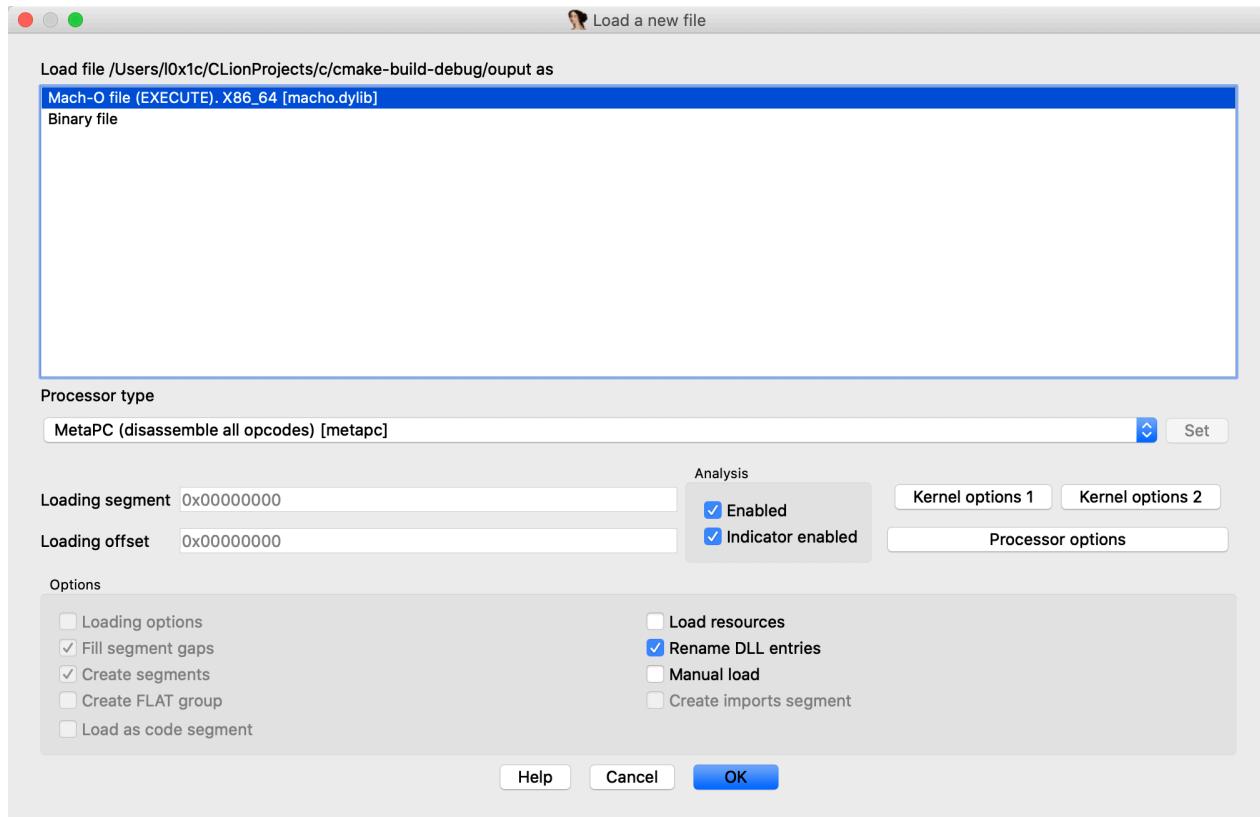
Reverse Engineering Tutorial with IDA Pro – An Introduction

ida的一个基本操作，一般来说我们的ida在网上都会有破解的版本，同时也有正版（如果家里有矿的话），在网上down一下即可



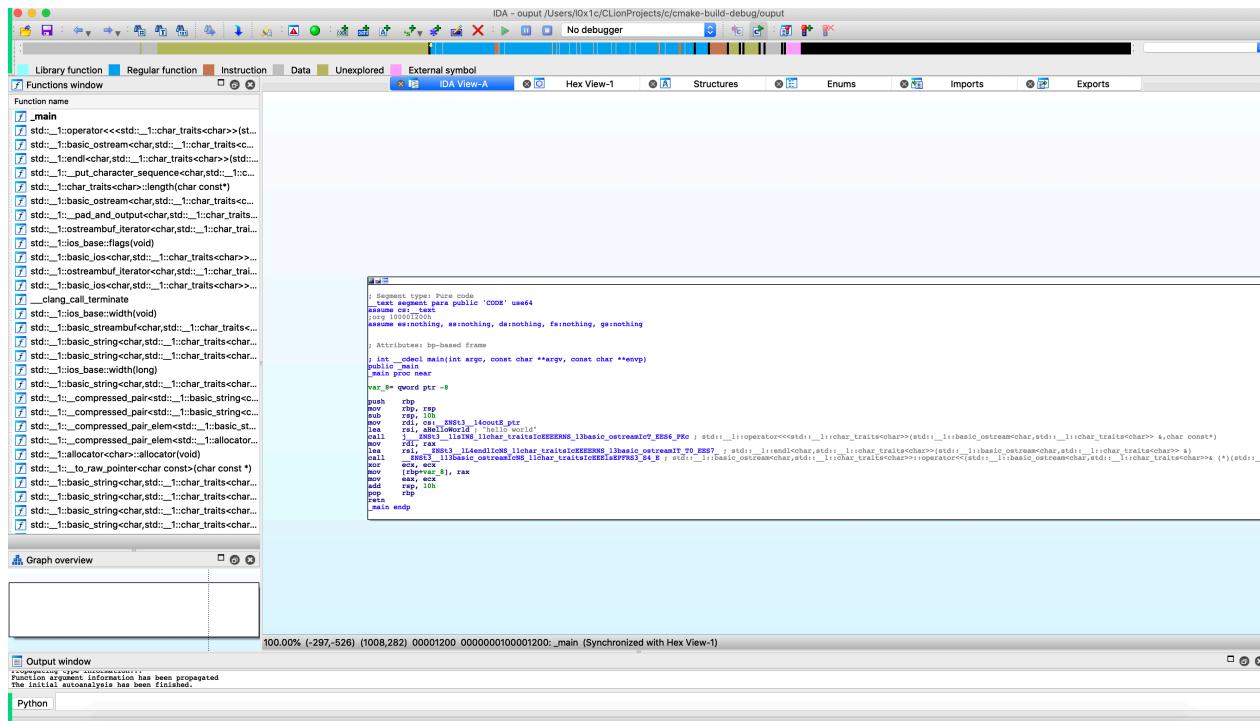
如果我们要打开一个新的文件的话，点击**new**即可，如果不喜欢看到这个欢迎的消息，直接点击我们的Display at startup取消即可，我们再启动的时候IDA会默认我们点击了**go**按钮，直接进入IDA的空白工作区，**previous**代表了我们打开最近用过的文件，IDA的注册History子项的值，最初的设定是10，也可以自己去设定

这里写了一个很简单的一个hello world我们用ida打开：

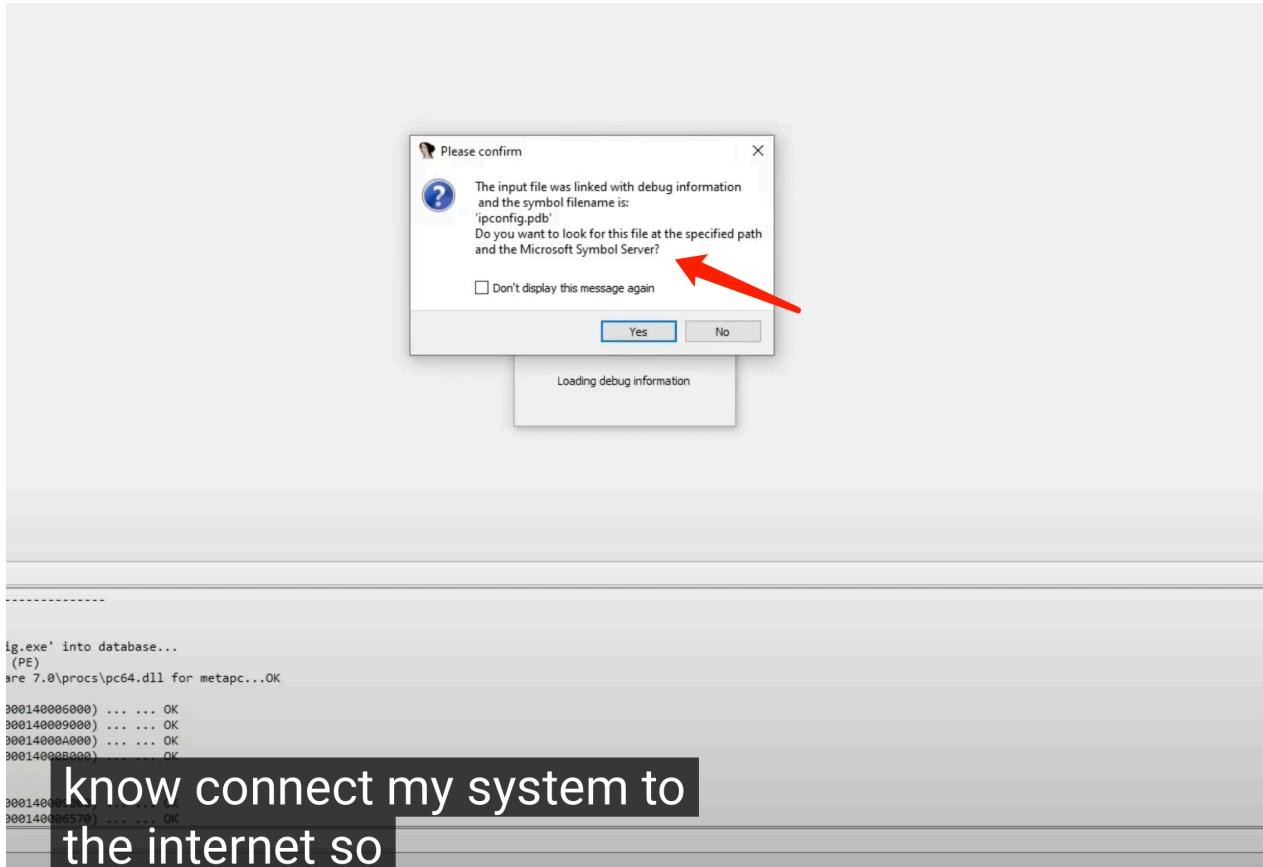


第一行应该知道是我们的加载器可以识别选定的文件，第二行是我们的Binary file，他会一直有的，这个是IDA加载无法识别文件的默认选项，提供了最低级的文件加载的方式

processor代表了IDA将根据从可执行文件的头中读取到的信息，选择合适的处理器，如果ida无法正确确定与所打开的文件关联的处理器类型，在文件加载之前需要我们手动选择一种处理器的类型



这是我们进入的页面，视频里有一个地方是联网进行获取的操作



官方给了相应的解释

此命令将加载PDB文件。

如果要反汇编的程序具有配套的PDB文件，则可以使用此命令将信息从PDB文件加载到数据库中。

请注意，PDB加载器使用系统DLL IMAGEHLP.DLL加载PDB文件。如果系统DLL是来自其他操作系统版本或出于某些其他原因，则它可能会拒绝加载PDB文件。

系统在系统目录（通常是\ WINNT \ Symbols之类）和当前目录中搜索PDB文件。

LoadPdbFile命令启动一个特殊的插件PDB.DLL，可以使用Edit-> Plugins子菜单手动运行它。此插件使用MS DIA DLL加载调试信息，因此只能加载与当前安装的IMAGEHLP.DLL兼容的PDB。

由于PDB.DLL使用仅在Windows上可用的MS DIA DLL，因此在Unix上加载PDB信息需要不同的设置。用户必须在Windows计算机上启动win32_remote.exe调试器服务器。Unix上的PDB插件可以连接到调试器服务器，并使用它来加载PDB文件。请注意，这里我们不调试任何东西，而是使用win32_remote.exe实用程序从PDB文件中提取信息并将其传输到IDA。为了确保一切正常，您必须提供以下内容：

-运行PDB.DLL或win32_remote.exe的计算机必须具有
安装的Microsoft调试工具。如有疑问，请验证
Microsoft的dia2dump.exe实用程序可以运行和加载PDB文件
在那台计算机上。

-如果IDA在Unix上运行，则必须有网络连接
在Unix计算机和运行win32_remote.exe的计算机之间。
您可以编辑pdb.cfg文件并指定远程
Windows计算机位于，因此您无需分别指定它

我们在窗口的左侧可以看到一些函数的名字以及函数的有趣的信息，列举IDA在数据库中识别的每一个函数

The screenshot shows the Functions window in IDA Pro. The left pane displays a list of function symbols with their segment, start address, length, locals, arguments, and registers. The right pane shows the assembly code for the selected function, with a red arrow pointing from the assembly code area towards the right edge of the window.

从左到右的功能：

- 函数名称

- 包含功能的细分
- 段内功能的偏移量
- 函数长度（以字节为单位）
- 局部变量+保存的寄存器的大小（以字节为单位）
- 传递给函数的参数的大小（以字节为单位）

该窗口的最后一列具有以下格式：

R- 函数返回给调用者

F- 远距功能

L- 库函数

S- 静态功能

B- 基于BP的帧。IDA会自动转换

所有要存储的帧指针[BP + XXX]操作数

变量。

T- 函数具有类型信息

= - 帧指针等于初始堆栈指针

在这种情况下，框架指针指向框架的底部

M- 保留

S- 保留

我- 保留

C- 保留

D- 保留

V- 保留

回到我们的代码的窗口

Text view:

The screenshot shows a debugger window with assembly code on the left and a context menu on the right. The assembly code is for a C program named 'main'.

```

; Segment type: Pure code
._text segment para public 'CODE' use64
assume cs: _text
org 100001200h
assume es:nothing, ss:nothing, ds:nothing, fs:nothing, gs:nothing

; Attributes: bp-based frame
; int __cdecl main(int argc, const char **argv, const c
public _main
_main proc near

var_8= qword ptr -8

push    rbp
mov     rbp, rsp
sub    rsp, 10h
mov     rdi, cs: _ZNSt3_14coutE_ptr
lea     rsi, aHelloWorld ; "hello world"
call    j__ZNSt3_11sINS_1lchar_traitsIcEEEERNS_13basic
mov     rdi, rax
lea     rsi, _ZNSt3_114endlICNS_1lchar_traitsIcEEEERN
call    __ZNSt3_113basic_ostreamICNS_11char_traitsIcEE
xor     ecx, ecx
mov     eax, ecx
add     rsp, 10h
pop     rbp
ret
_main endp

```

The context menu, titled 'Group nodes', is open over the assembly code. The 'Text view' option is highlighted with a blue background. A red arrow points from the bottom right towards the 'Text view' item.

- f Edit function...** ⌘P
- Set type...** Y
- Hide** ⌘+Numpad-
- Text view** **Text view** **Numpad-**
- Proximity browser** **Numpad-**
- U undefine** U
- Synchronize with** ►
- Add breakpoint** F2
- Xrefs graph to...**
- Xrefs graph from...**
- Font...**

```

100001200 ; ===== S U B R O U T I N E =====
100001200 ; Attributes: bp-based frame
100001200 ; int __cdecl main(int argc, const char **argv, const char **envp)
100001200 ; public _main
100001200 _main proc near
100001200
100001200 var_8= qword ptr -8
100001200
100001201 push    rbp
100001201 mov     rbp, rsp
100001201 sub    rsp, 10h
100001208 mov     rdi, cs: _ZNSt3_14coutE_ptr
100001208 lea     rsi, aHelloWorld ; "hello world"
100001208 call    j__ZNSt3_11sINS_1lchar_traitsIcEEEERNS_13basic
100001210 mov     rdi, rax
100001210 lea     rsi, _ZNSt3_114endlICNS_1lchar_traitsIcEEEERN
100001210 call    __ZNSt3_113basic_ostreamICNS_11char_traitsIcEE
100001212 xor     ecx, ecx
100001212 mov     eax, ecx
100001212 add     rsp, 10h
100001212 pop     rbp
100001212 ret
100001212 _main endp
100001237
100001237 align 20h
100001240 ; ===== S U B R O U T I N E =====
100001240 ; Attributes: bp-based frame
100001240 ; std::_l::basic_ostream<char, std::_l::char_traits<char>> & std::_l::operator<<(std::_l::basic_ostream<char, std::_l::char_traits<char>> &, char const*)
100001240 ; public _ZNSt3_11sINS_1lchar_traitsIcEEEERNS_13basic_ostreamICt_EES6_PKc ; weak
100001240 _ZNSt3_11sINS_1lchar_traitsIcEEEERNS_13basic_ostreamICt_EES6_PKc proc near
100001240 ; DATA XREF: la symbol ptr: _ZNSt3_11sINS_1lchar_traitsIcEEEERNS_13basic_ostreamICt_EES6_PKc ptr
100001240
100001240 var_20= qword ptr -20h
100001240 var_18= qword ptr -18h
100001240 var_10= qword ptr -10h
100001240 var_8= qword ptr -8
100001240
100001241 push    rbp
100001241 mov     rbp, rsp
100001241 sub    rsp, 20h
100001241 mov     [rbp+var_8], rdi
100001242 mov     [rbp+var_10], rsi
100001250 mov     rdi, [rbp+var_8]
100001250 mov     rsi, [rbp+var_10]
100001258 mov     rax, [rbp+var_10]
100001258 mov     [rbp+var_18], rdi
100001260 mov     rdi, rax
100001263 mov     [rbp+var_20], rsi
100001267 call    j__ZNSt3_11char_traitsIcE6lengthEPKc ; std::_l::char_traits<char>::length(char const*)
100001267 mov     rdi, [rbp+var_18]
100001267 mov     rsi, [rbp+var_20]
100001274 mov     rdx, rax
100001277 call    j__ZNSt3_124__put_character_sequenceICNS_1lchar_traitsIcEEEERNS_13basic_ostreamICt_TO_EES7_PKS4_m ; std::_l::__put_character_sequence<char, std::_l::char_traits<char>>(std::_l::bas
100001277 add     rsp, 20h
100001280 pop     rbp
100001281 ret
100001281 _ZNSt3_11sINS_1lchar_traitsIcEEEERNS_13basic_ostreamICt_EES6_PKc endp
100001281
100001281 align 10h
100001290
100001290 ===== S U B R O U T I N E =====

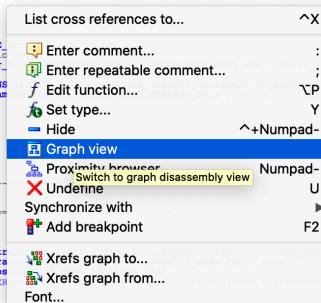
```

Graph view:

```

text:00000000100001200 ; ===== S U B R O U T I N E =====
text:00000000100001200 ; Segment type: Pure code
text:00000000100001200 ;   text    segment para public 'CODE' use64
text:00000000100001200 ;     assume  cs: text
text:00000000100001200 ;     assume  ss:nothing, ds:nothing, fs:nothing, gs:nothing
text:00000000100001200 ;     assume es:nothing, ss:nothing, ds:nothing, fs:nothing, gs:nothing
text:00000000100001200 ;     proc near
text:00000000100001200 ;       main
text:00000000100001200 ;           proc near
text:00000000100001200 ;             var_8 = qword ptr -8
text:00000000100001200 ;             push rbp
text:00000000100001201 ;             mov rbp, rsp
text:00000000100001202 ;             sub rbp, 10h
text:00000000100001203 ;             mov rdi, cs:_ZNSt3_I14coutE
text:00000000100001204 ;             mov rsi, _ZNS3_I14coutE
text:00000000100001205 ;             lea rdx, _ZNS3_I14coutE
text:00000000100001206 ;             mov rax, _ZNS3_I14coutE
text:00000000100001207 ;             xor eax, eax
text:00000000100001208 ;             mov eax, eax
text:00000000100001209 ;             add rax, 10h
text:0000000010000120A ;             pop rbp
text:0000000010000120B ;             ret
text:0000000010000120C ;             main
text:0000000010000120D ;           endp
text:0000000010000120E ; ===== S U B R O U T I N E =====
text:0000000010000120E ; Segment type: Pure code
text:0000000010000120E ;   text    segment para public 'CODE' use64
text:0000000010000120E ;     assume  cs: text
text:0000000010000120E ;     assume  ss:nothing, ds:nothing, fs:nothing, gs:nothing
text:0000000010000120E ;     assume es:nothing, ss:nothing, ds:nothing, fs:nothing, gs:nothing
text:0000000010000120E ;     proc near
text:0000000010000120E ;       main
text:0000000010000120E ;           proc near
text:0000000010000120E ;             align 20h
text:0000000010000120F ;             align 20h
text:0000000010000120F ;             main
text:0000000010000120F ;             proc near
text:0000000010000120F ;               var_20 = qword ptr -20h
text:0000000010000120F ;               var_18 = qword ptr -18h
text:0000000010000120F ;               var_10 = qword ptr -10h
text:0000000010000120F ;               var_8 = qword ptr -8
text:0000000010000120F ;               push rbp
text:0000000010000120F ;               mov rbp, rsp
text:0000000010000120F ;               sub rbp, 20h
text:0000000010000120F ;               mov rdi, [rbp+var_8], rdi
text:0000000010000120F ;               mov [rbp+var_10], rsi
text:0000000010000120F ;               mov rdi, [rbp+var_10], rsi
text:0000000010000120F ;               mov rax, [rbp+var_10], rax
text:0000000010000120F ;               mov [rbp+var_18], rdi
text:0000000010000120F ;               mov rdi, rax
text:0000000010000120F ;               mov rax, [rbp+var_20], rsi
text:0000000010000120F ;               call _ZNSt3_I14char_traitsIcEEEERN3_I3basic_ostreamIcT_EES6_Pk
text:0000000010000120F ;               std::operator<<(std::basic_ostream<char, std::char_traits<char>> &, char const*)
text:0000000010000120F ;               mov rdi, [rbp+var_18]
text:0000000010000120F ;               mov rax, [rbp+var_20]
text:0000000010000120F ;               mov rdx, rax
text:0000000010000120F ;               call _ZNSt3_I14put_character_sequenceIcNS_11char_traitsIcEEEERN3_I3basic_ostreamIT_T0_EES7_Pk4_m
text:0000000010000120F ;               std::operator<<(std::basic_ostream<char, std::char_traits<char>> &, char const*)
text:0000000010000120F ;               add rbp, 20h
text:0000000010000120F ;               pop rbp
text:0000000010000120F ;               ret
text:0000000010000120F ;             _ZNSt3_I11sINS_11char_traitsIcEEEERN3_I3basic_ostreamIcT_EES6_Pk
text:0000000010000120F ;           endp
text:0000000010000120F ;             align 10h
text:0000000010000120F ;             main
text:0000000010000120F ;           proc near
text:0000000010000120F ;             align 10h
text:0000000010000120F ;             main
text:0000000010000120F ;           endp
text:0000000010000120F ; ===== S U B R O U T I N E =====

```



```

; Segment type: Pure code
;   text    segment para public 'CODE' use64
;     assume cs: text
;     assume ss:nothing, ds:nothing, fs:nothing, gs:nothing
;
; Attributes: bp-based frame
; int __cdecl main(int argc, const char **argv, const char **envp)
; public main
;     proc near
; Var_8 = qword ptr -8
;     push rbp
;     rbp, rsp
;     sub rbp, 10h
;     mov rdi, cs:_ZNSt3_I14cout_ptr
;     rsi, _ZNS3_I14coutE
;     lea rdx, _ZNS3_I14coutE
;     call _ZNSt3_I14char_traitsIcEEEERN3_I3basic_ostreamIcT_EES6_Pk : std::operator<<(std::basic_ostream<char, std::char_traits<char>> &, char const*)
;     mov rdi, rax
;     rsi, _ZNS3_I14char_traitsIcEEEERN3_I3basic_ostreamIcT_EES7_Pk4_m : std::operator<<(std::basic_ostream<char, std::char_traits<char>> &, char const*)
;     call _ZNSt3_I14basic_ostreamIcNS_11char_traitsIcEEEERN3_I3basic_ostreamIcT_EES6_Pk : std::operator<<(std::basic_ostream<char, std::char_traits<char>> &, char const*)
;     xor eax, eax
;     mov eax, eax
;     add rbp, 10h
;     ret
;     main
;   endp

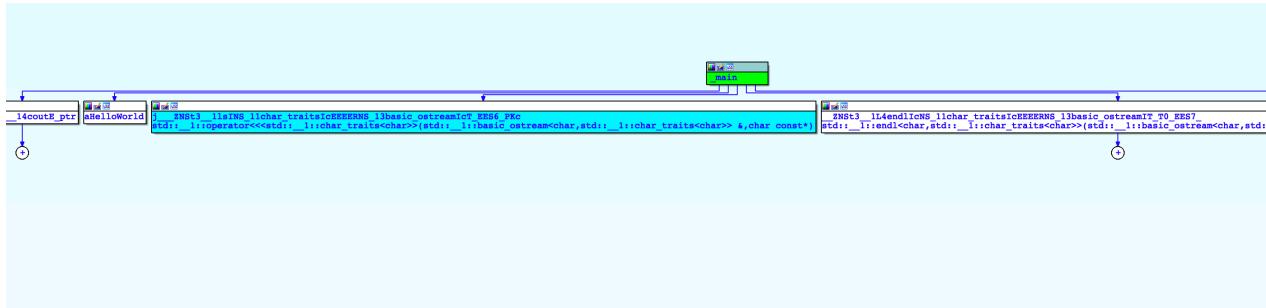
```

Proximity browser:

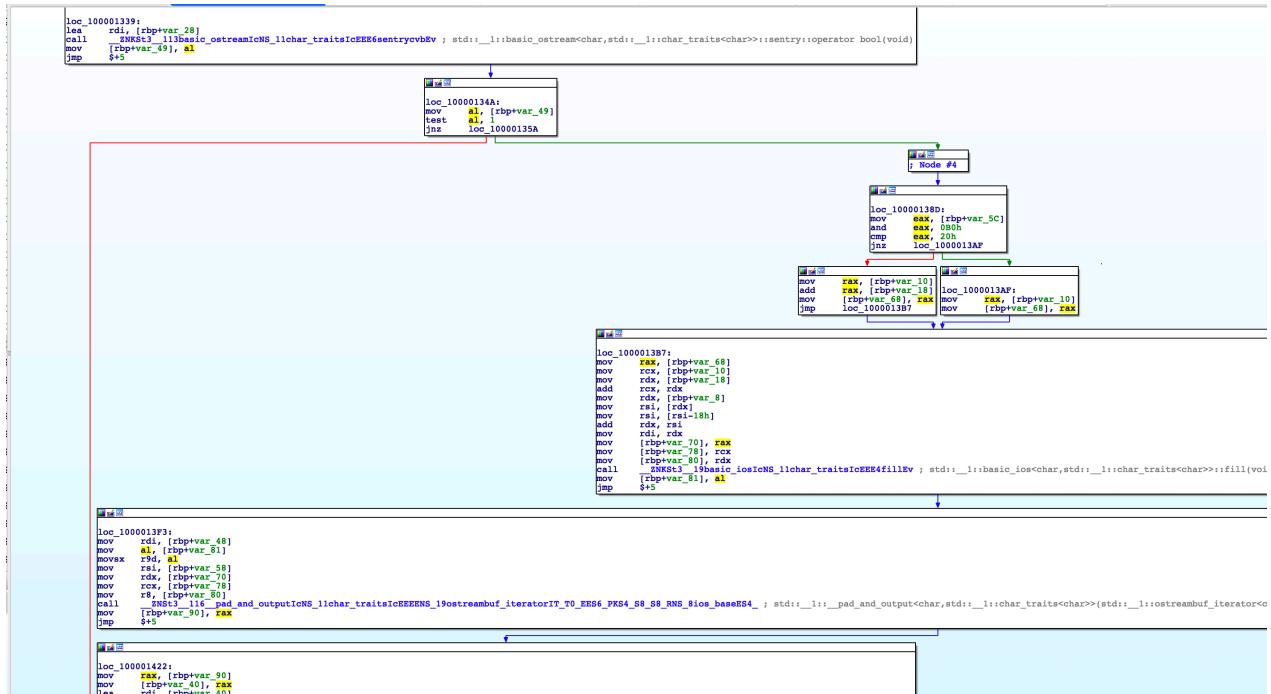
```

; Segment type: Pure code
;   text    segment para public 'CODE' use64
;     assume cs: text
;     assume ss:nothing, ds:nothing, fs:nothing, gs:nothing
;
; Attributes: bp-based frame
; int __cdecl main(int argc, const char **argv, const char **envp)
; public main
;     proc near
; Var_8 = qword ptr -8
;     push rbp
;     rbp, rsp
;     sub rbp, 10h
;     mov rdi, cs:_ZNSt3_I14cout_ptr
;     rsi, _ZNS3_I14coutE
;     lea rdx, _ZNS3_I14coutE
;     call _ZNSt3_I14char_traitsIcEEEERN3_I3basic_ostreamIcT_EES6_Pk : std::operator<<(std::basic_ostream<char, std::char_traits<char>> &, char const*)
;     mov rdi, rax
;     rsi, _ZNS3_I14char_traitsIcEEEERN3_I3basic_ostreamIcT_EES7_Pk4_m : std::operator<<(std::basic_ostream<char, std::char_traits<char>> &, char const*)
;     call _ZNSt3_I14basic_ostreamIcNS_11char_traitsIcEEEERN3_I3basic_ostreamIcT_EES6_Pk : std::operator<<(std::basic_ostream<char, std::char_traits<char>> &, char const*)
;     xor eax, eax
;     mov eax, eax
;     add rbp, 10h
;     ret
;     main
;   endp

```

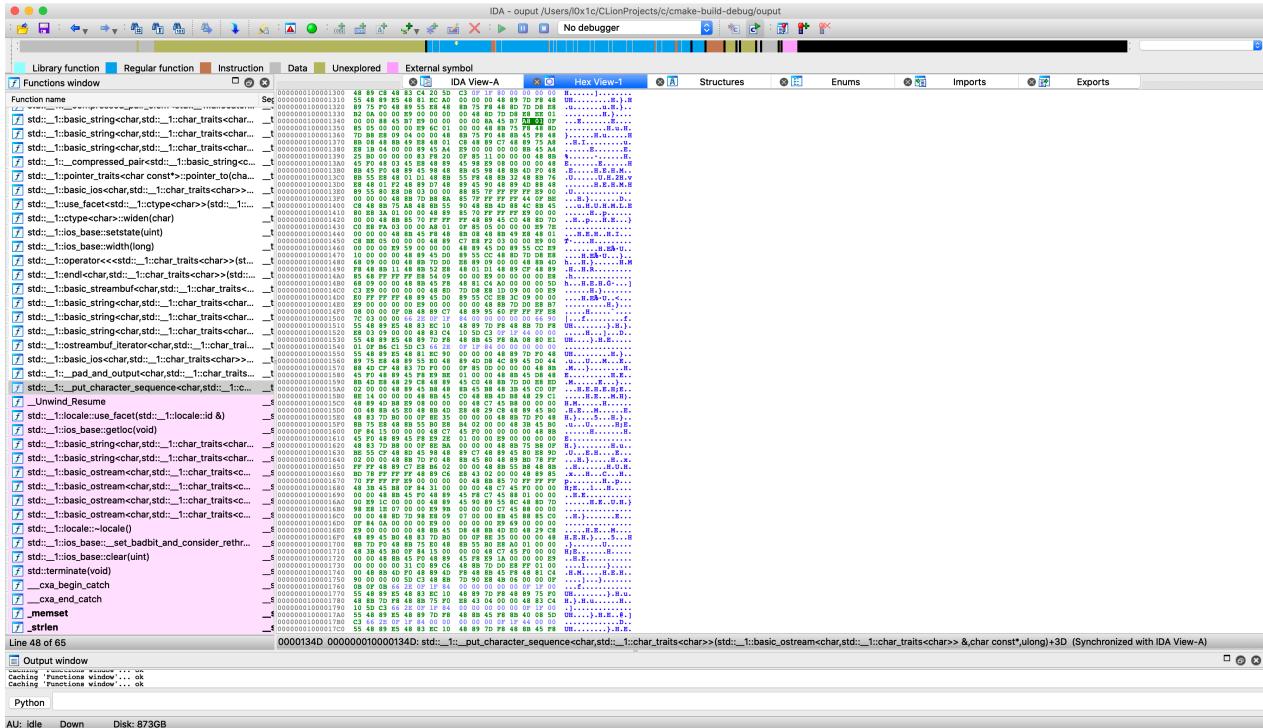


在图形模式下：

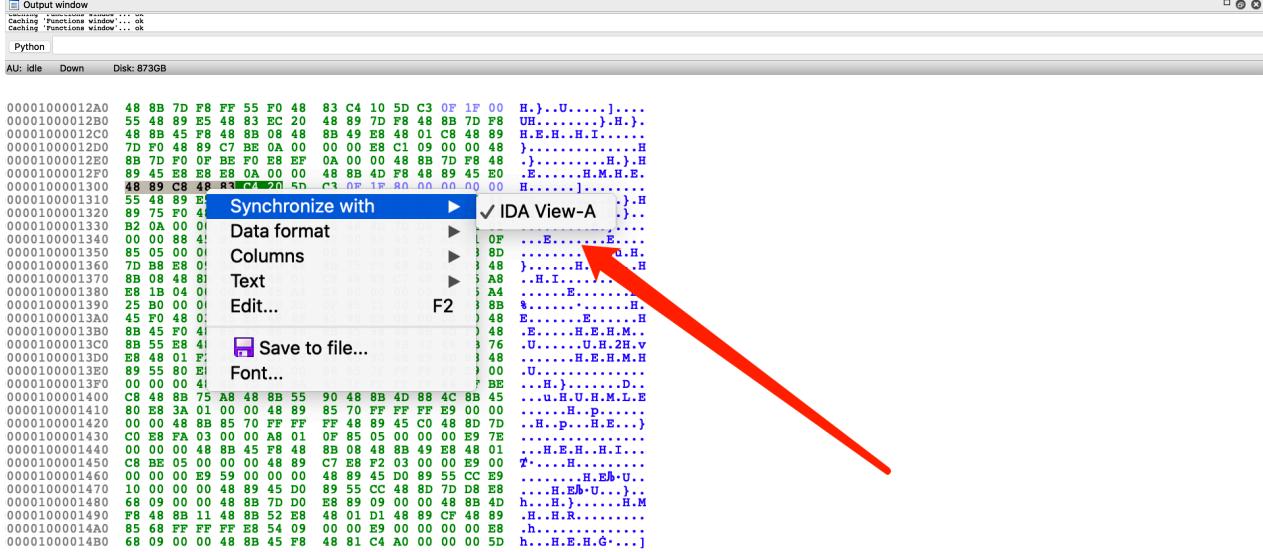


在条件跳转位置终止的基本块可能会产生两种流，YES边的检讨默认为绿色，NO边的箭头默认为红色，只有一个后继块的基本块会利用一个正常边默认为蓝色指向下一个即将执行的块

十六进制窗口：

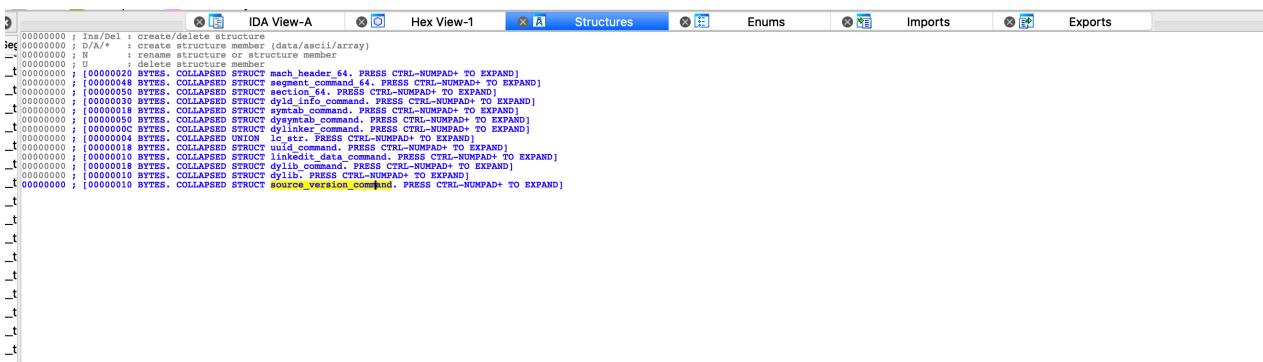


Line 48 of 65 0000134D 000000010000134D: std::put_character_sequence<char, std::char_traits<char>>(std::basic_ostream<char, std::char_traits<char>>, char const*, ulong) + 3D (Synchronized with IDA View-A)



和ida view-A对应，一般来说如果我们的hex窗口中显示的全部都是问号，表示ida无法识别给定的虚拟地址范围内的值，如果程序中包含一个bss节，因为bss不占用文件的空间，但是加载会扩展这一节，以适应程序的静态存储要求

结构体窗口：



为标准数据结构布局提供现成的参考

枚举窗口：

如果线程到枚举数据类型，将在枚举窗口中列出该数据类型，也可以自己枚举代替整数常量

导入窗口：

Address	Ordinal	Name	Library
00000000...		std::__1::cout	/usr/lib/libc+++.1.dylib
00000000...		std::__1::ctype<char>::id	/usr/lib/libc+++.1.dylib
00000000...		_gxx_personality_v0	/usr/lib/libc+++.1.dylib
00000000...		std::__1::locale::use_facet(std::__1::locale::id &)	/usr/lib/libc+++.1.dylib
00000000...		std::__1::ios_base::getloc(void)	/usr/lib/libc+++.1.dylib
00000000...		std::__1::basic_string<char, std::__1::char_traits<char>, std::__1::allocator<char>>::basic_string(char_type, size_type, const Allocator &)	/usr/lib/libc+++.1.dylib
00000000...		std::__1::basic_string<char, std::__1::char_traits<char>, std::__1::allocator<char>>::basic_string(const char_type*, size_type, const Allocator &)	/usr/lib/libc+++.1.dylib
00000000...		std::__1::basic_ostream<char, std::__1::char_traits<char>, std::__1::allocator<char>>::basic_ostream(char_type, const Allocator &)	/usr/lib/libc+++.1.dylib
00000000...		std::__1::basic_ostream<char, std::__1::char_traits<char>, std::__1::allocator<char>>::basic_ostream(const char_type*, const Allocator &)	/usr/lib/libc+++.1.dylib
00000000...		std::__1::basic_ostream<char, std::__1::char_traits<char>, std::__1::allocator<char>>::operator<<(const char_type*, const Allocator &)	/usr/lib/libc+++.1.dylib
00000000...		std::__1::basic_ostream<char, std::__1::char_traits<char>, std::__1::allocator<char>>::operator<<(const Allocator &)	/usr/lib/libc+++.1.dylib
00000000...		std::__1::ios_base::~set_badbit_and_consider_rethrow(...)	/usr/lib/libc+++.1.dylib
00000000...		std::__1::ios_base::clear(uint)	/usr/lib/libc+++.1.dylib
00000000...		std::terminate(void)	/usr/lib/libc+++.1.dylib
00000000...		__cxa_begin_catch	/usr/lib/libc+++.1.dylib
00000000...		__cxa_end_catch	/usr/lib/libc+++.1.dylib
00000000...		dyld_stub_binder	/usr/lib/libSystem.B.dylib
00000000...		_Unwind_Resume	/usr/lib/libSystem.B.dylib
00000000...		_memset	/usr/lib/libSystem.B.dylib
00000000...		_strlen	/usr/lib/libSystem.B.dylib

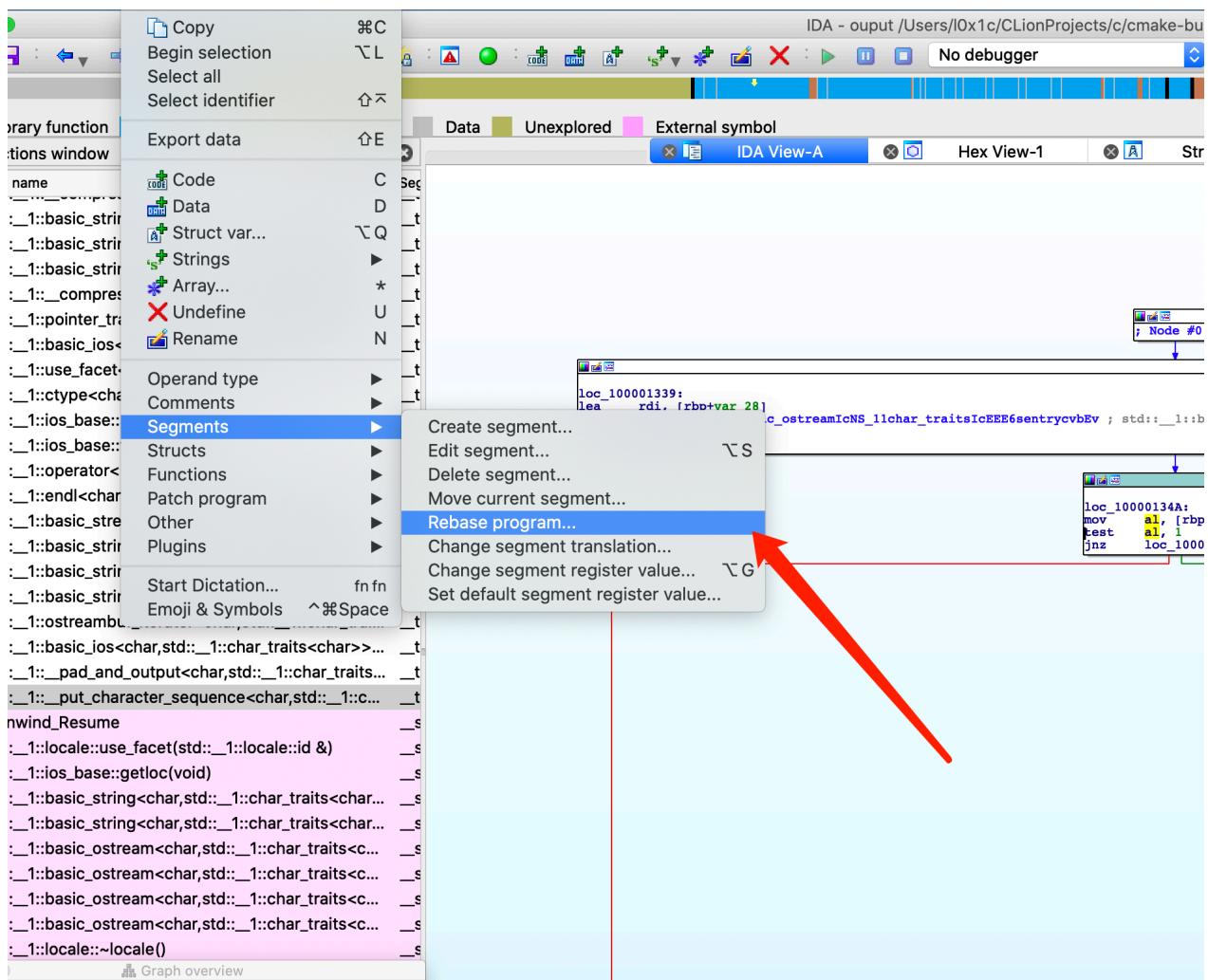
列出被分析的二进制导入的函数

导出窗口：

Name	Address	Ordinal
__mh_execute_header	0000000100000000	
std::__1::operator<<(std::__1::char_traits<char> &, std::__1::basic_ostream<char, std::__1::char_traits<char>, std::__1::allocator<char> &)	0000000100001240	
std::__1::put_character_sequence<char, std::__1::char_traits<char>, std::__1::allocator<char> >	0000000100001310	
std::__1::char_traits<char>::length(char const*)	0000000100001510	
std::__1::char_traits<char>::eq_int_type(int,int)	0000000100001C70	
std::__1::char_traits<char>::eof(void)	0000000100001C90	
__main	0000000100001200	[main entry]

列出文件的入口点，导出窗口至少包含一个项目：程序的执行入口点

一般喜欢用od，以及xdb进行调试的朋友，如果感觉基址感觉没有对上不太好观看的话可以修改这里





Rebase the whole program

Please enter the new



Address of the first segment



Shift delta



Image base

Value

0x100000000



Fix up the program



Rebase the whole image

Help

Cancel

OK

直接修改成相同的就好了，这样就可以相对应上了，一般一个小操作就是

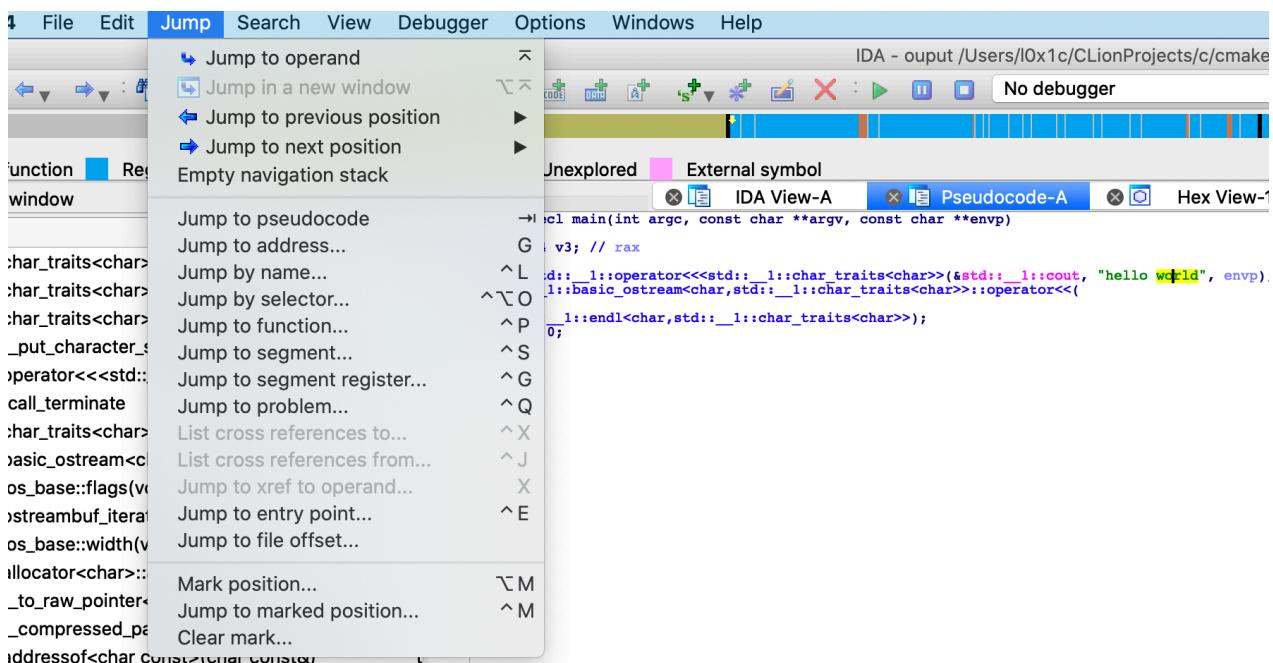
The screenshot shows the IDA Pro interface with the assembly code for the `main` function. The assembly code is as follows:

```
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     _int64 v3; // rax
4
5     v3 = std::operator<<<std::cout << "hello world", envp);
6     std::basic_ostream<char,std::char_traits<char>>::operator<<(v3,
7         std::endl<char,std::char_traits<char>>);
8
9     return 0;
0 }
```

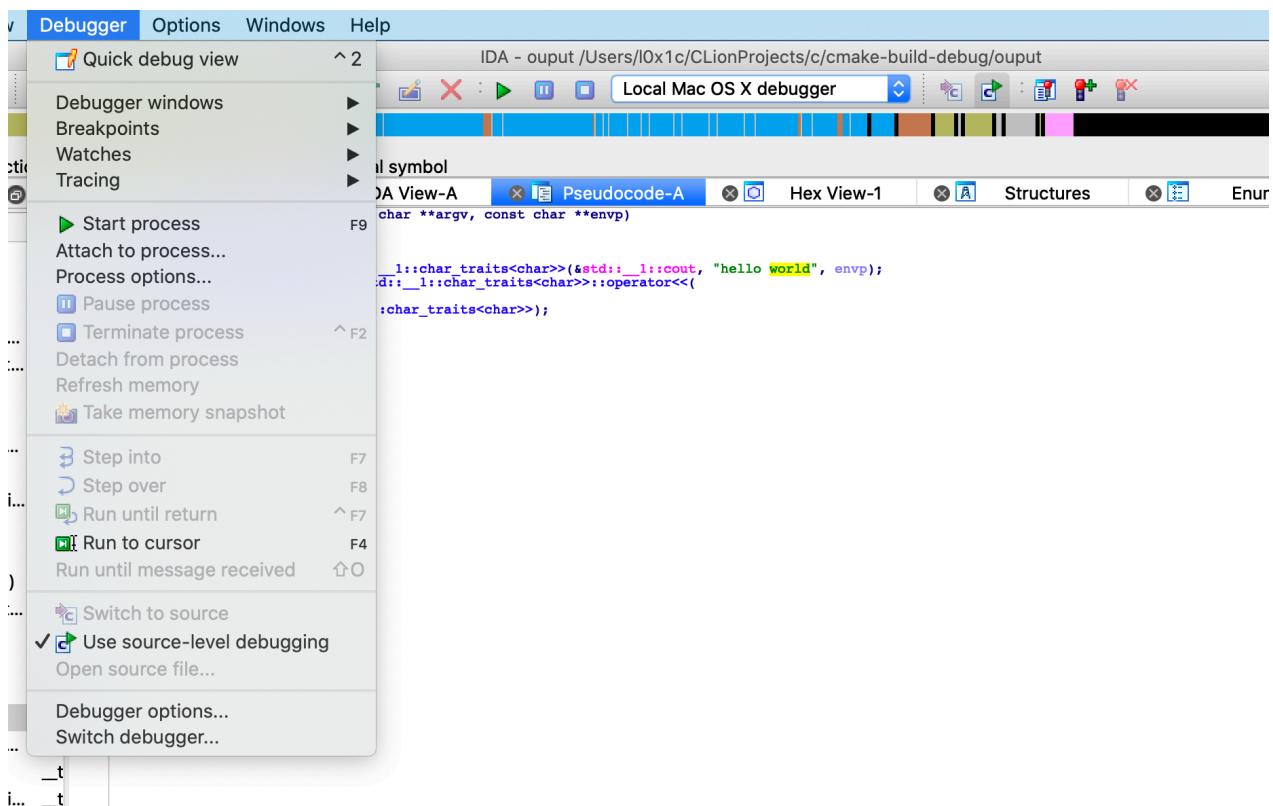
The address of the `main` function is highlighted in red at the bottom of the assembly window: `00000120F _main:5 10000120F`.

在下面的地方可以看到相应的地址位置，在od以及xdbg进行调试的时候，可以看到相应的位置

如果想跳转到各种位置，以及函数名等等的，可以直接Jump选择跳转



IDA也给我们了一个Debug的功能

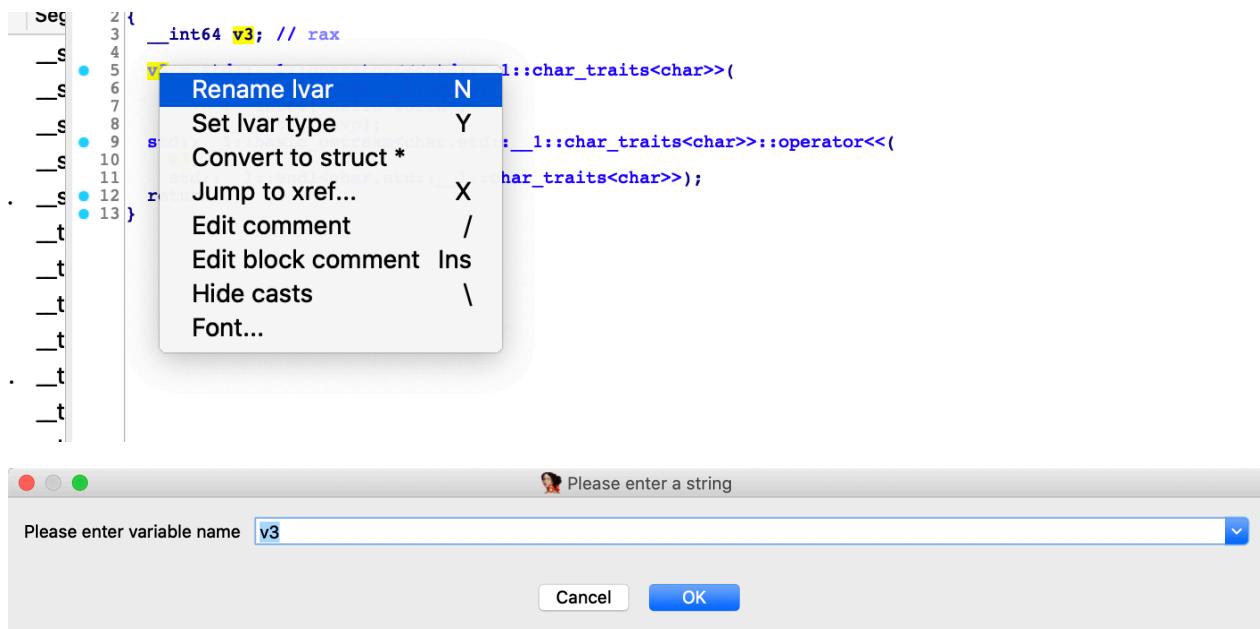


开始调试，下段点等等的操作都有

Reverse Engineering with IDA Pro – Fixing/Rebuilding Structure/Structs (Pseudocode)

这一节主要讲述了我们的一个f5的操作以及修改成员名字以及，结构体的一些使用

修改成员名字：



假设修改成struct1

IDA View-A | Pseudocode-C | Pseudocode-B | Pseudocode-A

```

1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     _int64 struct1; // rax
4
5     struct1 = std::__1::operator<<(std::__1::char_traits<char>)(
6         (_int64)&std::__1::cout,
7         (_int64)"Hello world",
8         (_int64)envp);
9     std::__1::basic_ostream<char,std::__1::char_traits<char>>::operator<<
0     struct1,
1     std::__1::endl<char,std::__1::char_traits<char>>;
2     return 0;
3 }

```

可以看到已经修改成功了

创建结构体：

IDA View-A | Pseudocode-B | Pseudocode-A | Hex View-1 | Structures | Enums | Imports

```

00000000 ; Ins/Del : create/delete structure
00000000 ; D/A/* : create structure member (data/ascii/array)
00000000 ; N : rename structure or structure member
00000000 ; U : delete structure member
00000000 ; [00000020 BYTES. COLLAPSED STRUCT mach_header_64. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000040 BYTES. COLLAPSED STRUCT segment_command_64. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000060 BYTES. COLLAPSED STRUCT section_64. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000080 BYTES. COLLAPSED STRUCT dyld_info_command. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000010 BYTES. COLLAPSED STRUCT dyld_command. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000018 BYTES. COLLAPSED STRUCT dylib_command. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000050 BYTES. COLLAPSED STRUCT dyntab_command. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000052 BYTES. COLLAPSED STRUCT dylinker_command. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000004 BYTES. COLLAPSED UNION lc_str. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000010 BYTES. COLLAPSED STRUCT linkedit_data_command. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000018 BYTES. COLLAPSED STRUCT dylib. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000010 BYTES. COLLAPSED STRUCT source_version_command. PRESS CTRL-NUMPAD+ TO EXPAND]

```

这里面可以自己去定义一些结构体，新建结构体，成员的大小，重命名成员，删除成员，删除结构体等等的操作

```

00000000 ; Ins/Del : create/delete structure
00000000 ; D/A/* : create structure member (data/ascii/array)
00000000 ; N : rename structure or structure member
00000000 ; U : delete structure member
00000000 ; [00000020 BYTES. COLLAPSED STRUCT mach_header_64. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ;
00000000 struct1      struc ; (sizeof=0x4, mappedto_26)
00000000 one          dd ?
00000004 struct1      ends
00000004
00000000 ; [00000048 BYTES. COLLAPSED STRUCT segment_command_64. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000050 BYTES. COLLAPSED STRUCT section_64. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000030 BYTES. COLLAPSED STRUCT dyld_info_command. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000018 BYTES. COLLAPSED STRUCT dyld_command. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000050 BYTES. COLLAPSED STRUCT dyntab_command. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000004 BYTES. COLLAPSED STRUCT dylinker_command. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000010 BYTES. COLLAPSED UNION lc_str. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000018 BYTES. COLLAPSED STRUCT uid_command. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000010 BYTES. COLLAPSED STRUCT linkedit_data_command. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000018 BYTES. COLLAPSED STRUCT dylib_command. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000010 BYTES. COLLAPSED STRUCT dylib. PRESS CTRL-NUMPAD+ TO EXPAND]
00000000 ; [00000010 BYTES. COLLAPSED STRUCT source_version_command. PRESS CTRL-NUMPAD+ TO EXPAND]

```

我们定义一个struct1的结构体，里面具有one的成员

IDA View-A | Pseudocode-A | Pseudocode-C | Hex View-1 | Structures

```

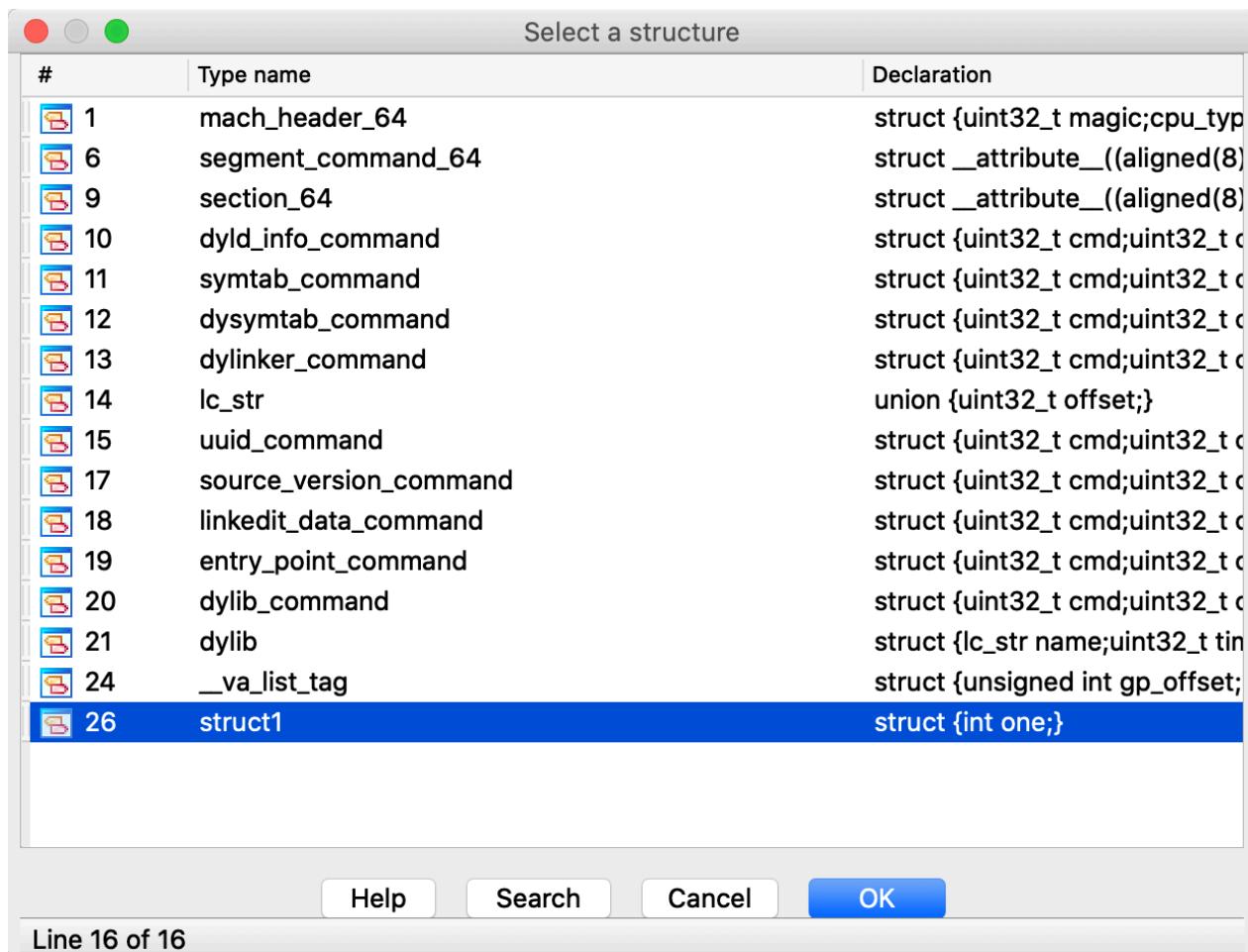
1 int __cdecl main(int argc, const char **argv, const char **envp)
2 {
3     _int64 struct1; // rax
4
5     std::operator<<(std::cout, struct1);
6     std::cout << std::endl;
7
8     return 0;
9 }
10
11
12
13

```

Context menu for line 5:

- Rename lvar N
- Set lvar type Y
- Convert to struct ***
- Jump to xref... X
- Edit comment /
- Edit block comment Ins
- Hide casts \
- Font...

我们链接一下结构体，看一下



可以看到结构体的位置被修改了，自己就是演示一下这个操作，具体的效果看一下这个视频里的我截取出来了：

```

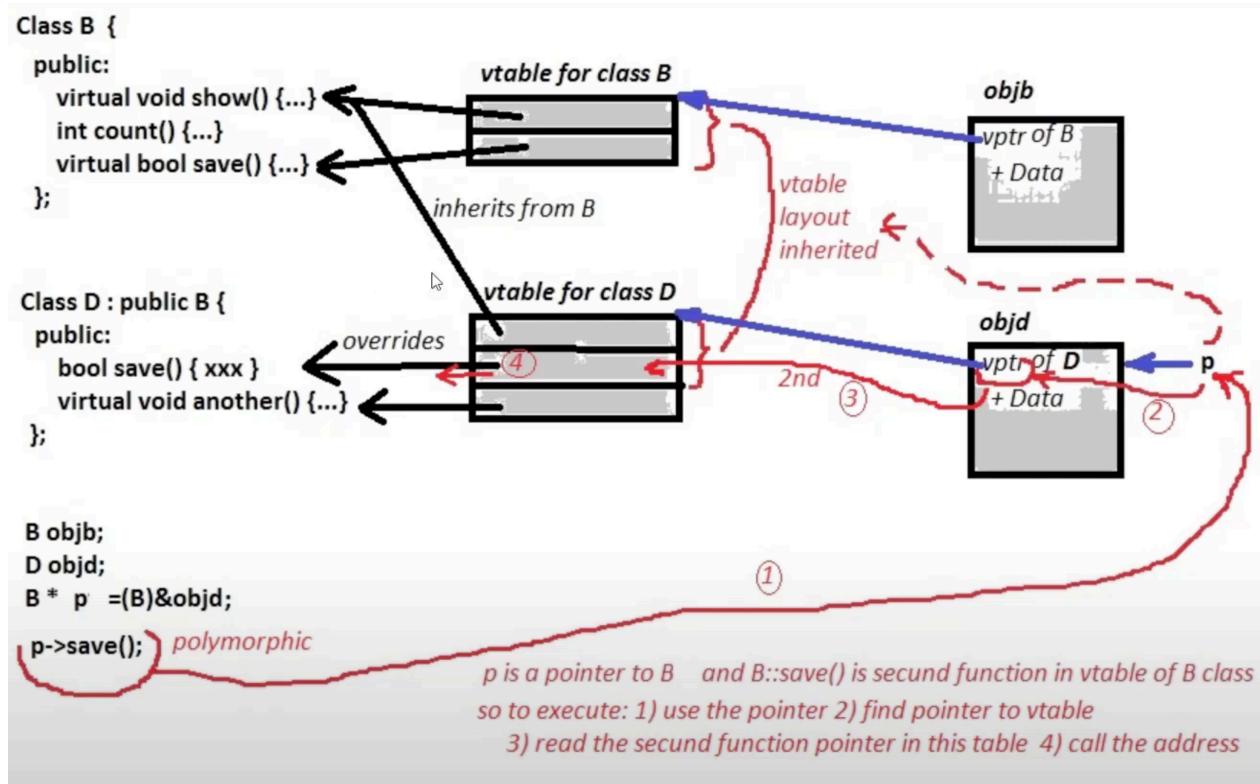
1 int64 sub_1400010A0()
2 {
3     mystruct1 *ptr_to_struct1; // rbx
4     _QWORD *v1; // rax
5     _QWORD v2; // rdi
6
7     ptr_to_struct1 = (mystruct1 *)malloc(0x20ui64);
8     v1 = malloc(0x10ui64);
9     ptr_to_struct1->myint1 = 1;
10    ptr_to_struct1->myint2 = 2;
11    v2 = v1;
12    ptr_to_struct1->myfloat1 = 0x40400000;
13    sub_140001010("Val : %d \n", 1i64);
14    ptr_to_struct1[1].myint1 = 1095520072;
15    *(QWORD *)&ptr_to_struct1[1].myint2 = sub_140001070;
16    *(QWORD *)&ptr_to_struct1[2].myint1 = v2;
17    *v2 = sub_140001080;
18    v2[1] = sub_140001090;
19    sub_140001080(1234164);
20    ((void (_fastcall *)(signed __int64))v2[1])(5678i64);
21    return 0i64;
22}

```

这里就是结构体修改后的结果，以前是一个数组的一个形式

C++ Reverse Engineering with IDA Pro - Rebuilding virtual function table (vftable)

这一节主要去让我们快速去找到虚函数表的一个方法插件



代码写一下测试一下：

```

main.cpp
1 #include <iostream>
2 using namespace std;
3
4 class base{
5 public:
6     virtual void print(){
7         cout << "print base class" << endl;
8     }
9     virtual void print1(){
10        cout << "print1 base class" << endl;
11    }
12    void show(){
13        cout << "show base class" << endl;
14    }
15};
16
17 class derived : public base{
18 public:
19     void print(){
20         cout << "print derived class" << endl;
21     }
22     void show(){
23         cout << "show derived class" << endl;
24     }
25};
26
27 int main()
28 {
29     derived d;
30     d.print();
31     d.show();
32     base b;
33     b.print();
34     b.show();
35 }

```

我们去ida看一下：

```

IDA - c /Users/0x1c/CLionProjects/c/cmake-build-debug/c
No debugger

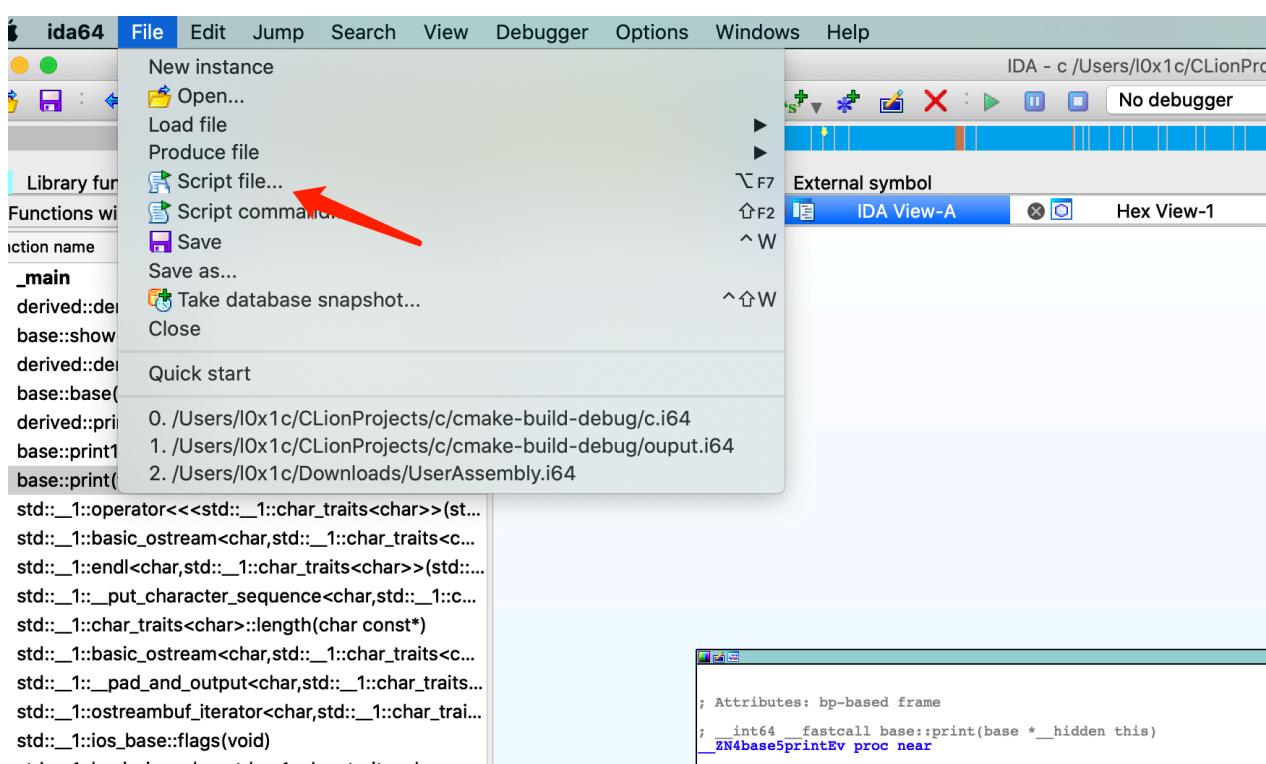
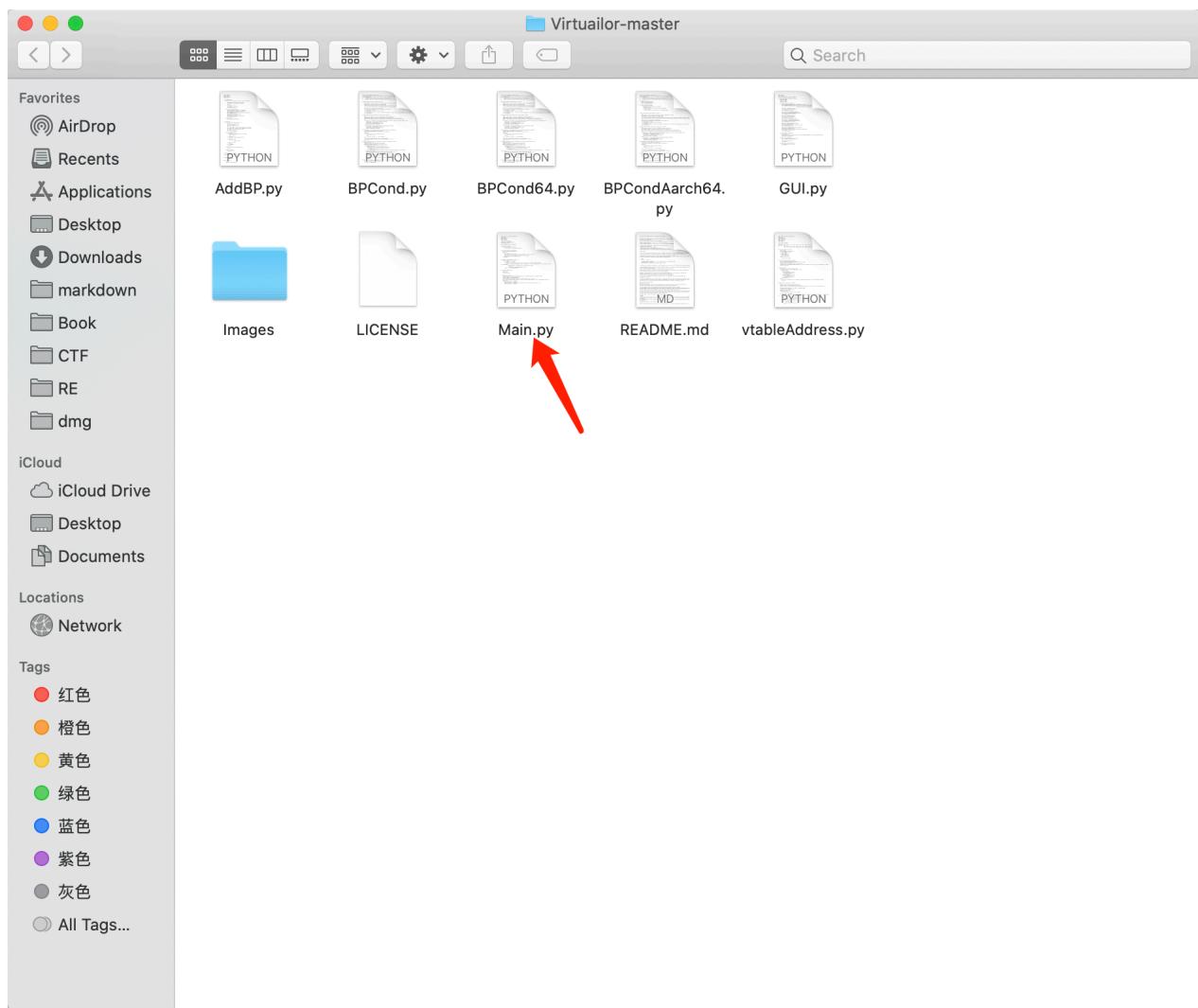
Functions window
Library function Regular function Instruction Data Unexplored External symbol
Functions window
Function name
_main
derived::derived(void)
base::show(void)
derived::derived(void)
base::base(void)
derived::print(void)
base::print1(void)
base::print(void)
std::__1::operator<<(std::__1::char_traits<char>)(st...
std::__1::basic_ostream<char, std::__1::char_traits<char>, ...
std::__1::endl<char, std::__1::char_traits<char>>(std::__1::...
std::__1::put_character_sequence<char, std::__1::char_trai...
std::__1::char_traits<char>::length(char const*)
std::__1::basic_ostream<char, std::__1::char_traits<char>...
std::__1::basic_ostream<char, std::__1::char_traits<char>...
std::__1::basic::pad_and_output<char, std::__1::char_traits<...
std::__1::basic::ostreambuf_iterator<char, std::__1::char_trai...
std::__1::ios_base::flags(void)
std::__1::basic::ios<char, std::__1::char_traits<char>...
std::__1::basic::ios<char, std::__1::char_traits<char>...
std::__1::basic::ios<char, std::__1::char_traits<char>...
__clang_call_terminate
std::__1::ios_base::width(void)
std::__1::basic::strmabuf<char, std::__1::char_traits<char>...
std::__1::basic::string<char, std::__1::char_traits<char>...
std::__1::basic::string<char, std::__1::char_traits<char>...
std::__1::ios_base::width(ong)
std::__1::basic::string<char, std::__1::char_traits<char>...
std::__1::compressed_pair<std::__1::basic_string<char, std::__1::...
std::__1::compressed_pair<std::__1::basic_string<char, std::__1::...
std::__1::compressed_pair<std::__1::basic_string<char, std::__1::...
Line of 73
Graph overview
100.00% (-280,-583) (876,332) 000001B0 00000001000001B0: base::print(void) (Synchronized with Hex View-1)

Output window
1000001260: using guessed type _int32 _fastcall std::__1::basic::ostream<char, std::__1::char_traits<char>>::operator<<(_int32 const char*, _int32);
100001260: using guessed type _int32 _fastcall std::__1::basic::ostream<char, std::__1::char_traits<char>>::operator<<(_int32 const char*, _int32, _int32);
100001260: using guessed type _int32 _fastcall std::__1::basic::ostream<char, std::__1::char_traits<char>>::operator<<(_int32 const char*, _int32, _int32, _int32);

Python
AU: idle Down Disk: 87GB

```

我们使用插件的话可以把虚函数表很好的在结构体里展示出来：<https://github.com/0xgalz/Virtualor>





这个main.py我们需要修改一下

```

26     return addr
27
28
29 def add_bp_to_virtual_calls(cur_addr, end):
30     while cur_addr < end:
31         if cur_addr == idc.BADADDR:
32             break
33         elif idc.print_insn_mnem(cur_addr) == 'call' or
34             idc.print_insn_mnem(cur_addr) == 'BLR':
35             if True in [idc.print_operand(cur_addr, 0).find(reg) != -1 for reg in
36                         REGISTERS]: # idc.GetOpnd(cur_addr, 0) in REGISTERS:
37                 cond, bp_address = vtableAddress.write_vtable2file(cur_addr)
38                 if cond != '':
39                     bp_vtable = AddBP.add(bp_address, cond)
40             cur_addr = idc.next_head(cur_addr)
41
42     def set_values(start, end):
43         start = start
44         end = end
45         return start, end
46
47 if __name__ == '__main__':
48     start_addr_range = idc.MinEA() # You can change the virtual calls address range
49     end_addr_range = idc.MaxEA()
50     oldTo = idaapi.set_script_timeout(0)
51     # Initializes the GUI: Deletes the 0x in the beginning and the L at the end:
52     gui = GUI.VirtualilorBasicGUI(set_values, {'start': hex(start_addr_range)[2:], 'end': hex(end_addr_range)[2:]})
53     gui.exec_()
54     if gui.start_line.text != "banana":
55         print("Virtualilor - Started")
56         add_bp_to_virtual_calls(int(gui.start_line.text(), 16),
57                                int(gui.stop_line.text(), 16))
58     print("Virtualilor - Finished")
59

```

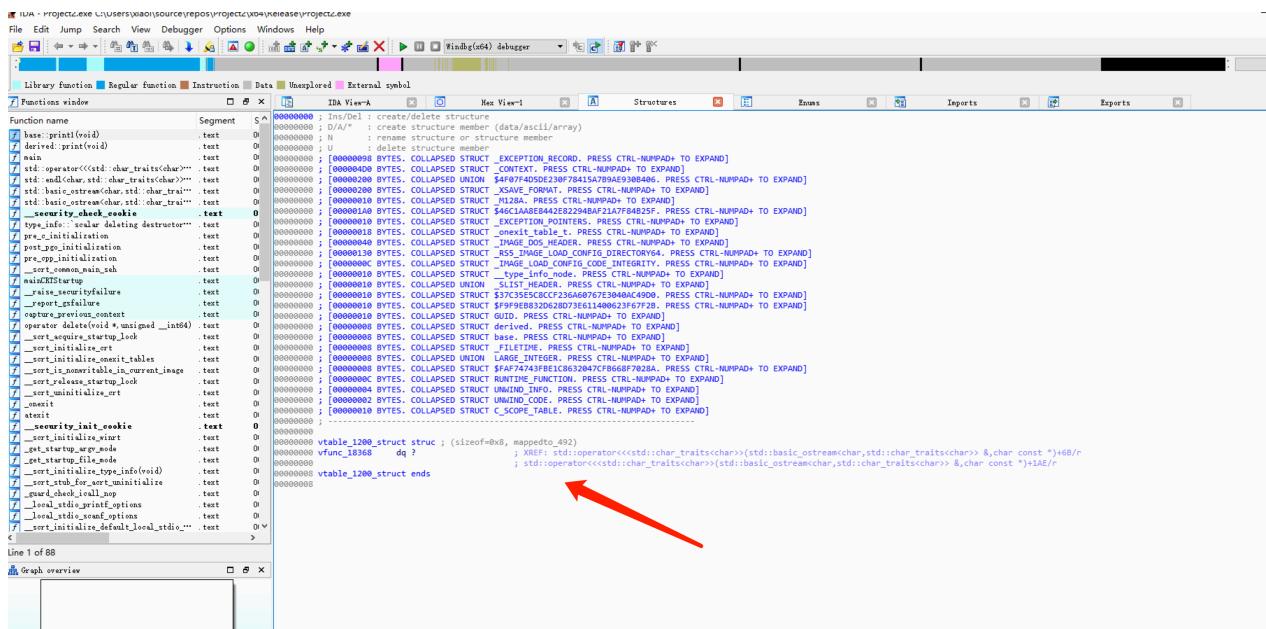
我们运行脚本后可看到成功了

```

FILE '/Users/0x1c/GIGIProjects/C/maker-build-debug/G' has been successfully loaded into the database.
IDA is analyzing the input file...
You may start to explore the input file right now.
Hex dump, dump plugins have been loaded (v1.0.0.170914)
Licenses: 55-3341-444-57 Jiang Tint, Personal license (1 user)
The program has been successfully loaded.
Please check the Edit/Plugins menu for more information.
IDA Pro 6.9.0.0.20190710.170914
Python 2.7.16 (default, Jul 5 2020, 02:24:03)
[GCC 4.2.1 Compatible Apple LLVM 11.0.3 (clang-1103.3.29.21) (-macos10.15-ojb)
IDA Pro 6.9.0.0.20190710.170914
Python 2.7.16 (default, Jul 5 2020, 02:24:03)
[GCC 4.2.1 Compatible Apple LLVM 11.0.3 (clang-1103.3.29.21) (-macos10.15-ojb)
Propagating type information...
Function argument information has been propagated
The function argument information has been propagated
Traceback (most recent call last):
  File "/Users/0x1c/RE/Virtualilor-master/GUI.py", line 53, in on_button_clicked
    start = int(self.start_line.text(), 16)
ValueError: invalid literal for int() with base 16: '10000000000L'
Traceback (most recent call last):
  File "/Users/0x1c/RE/Virtualilor-master/GUI.py", line 53, in on_button_clicked
    start = int(self.start_line.text(), 16)
ValueError: invalid literal for int() with base 16: '10000000000L'
WARNING: This program must be run as unsigned or run as root to debug mac applications.
Virtualilor - Finished

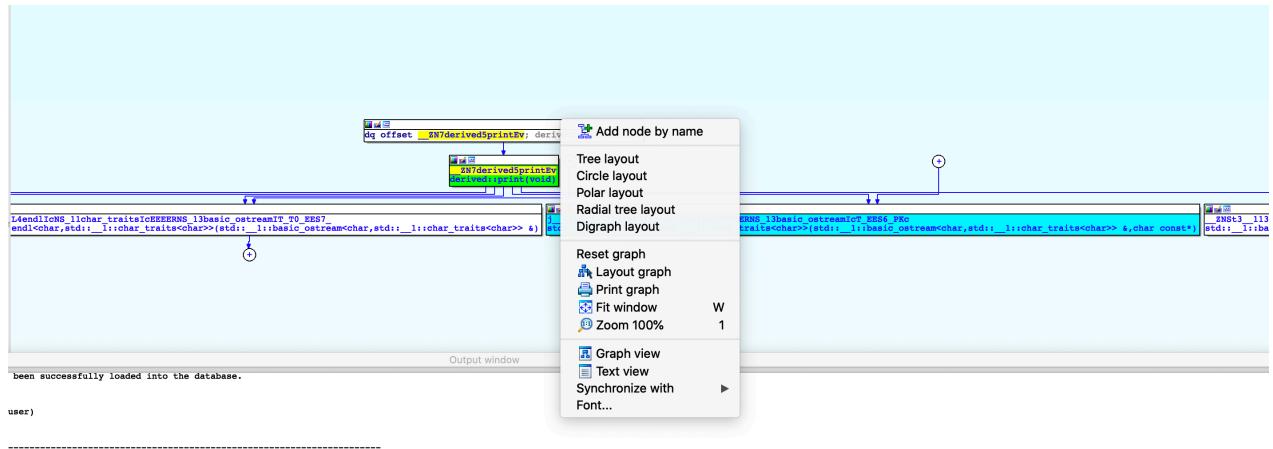
```

我们local windbg运行后可以直接看到虚函数表:



Reverse Engineering Tutorial with IDA Pro – Function Cross Reference & Proximity Browser

主要介绍了交叉引用，流程图的几种，演示一下：



可以根据 Tree layout那五个类似的去改变流程图的一个观看方式，如果要查看一个函数是谁调用的 (x/ctrl+x) :

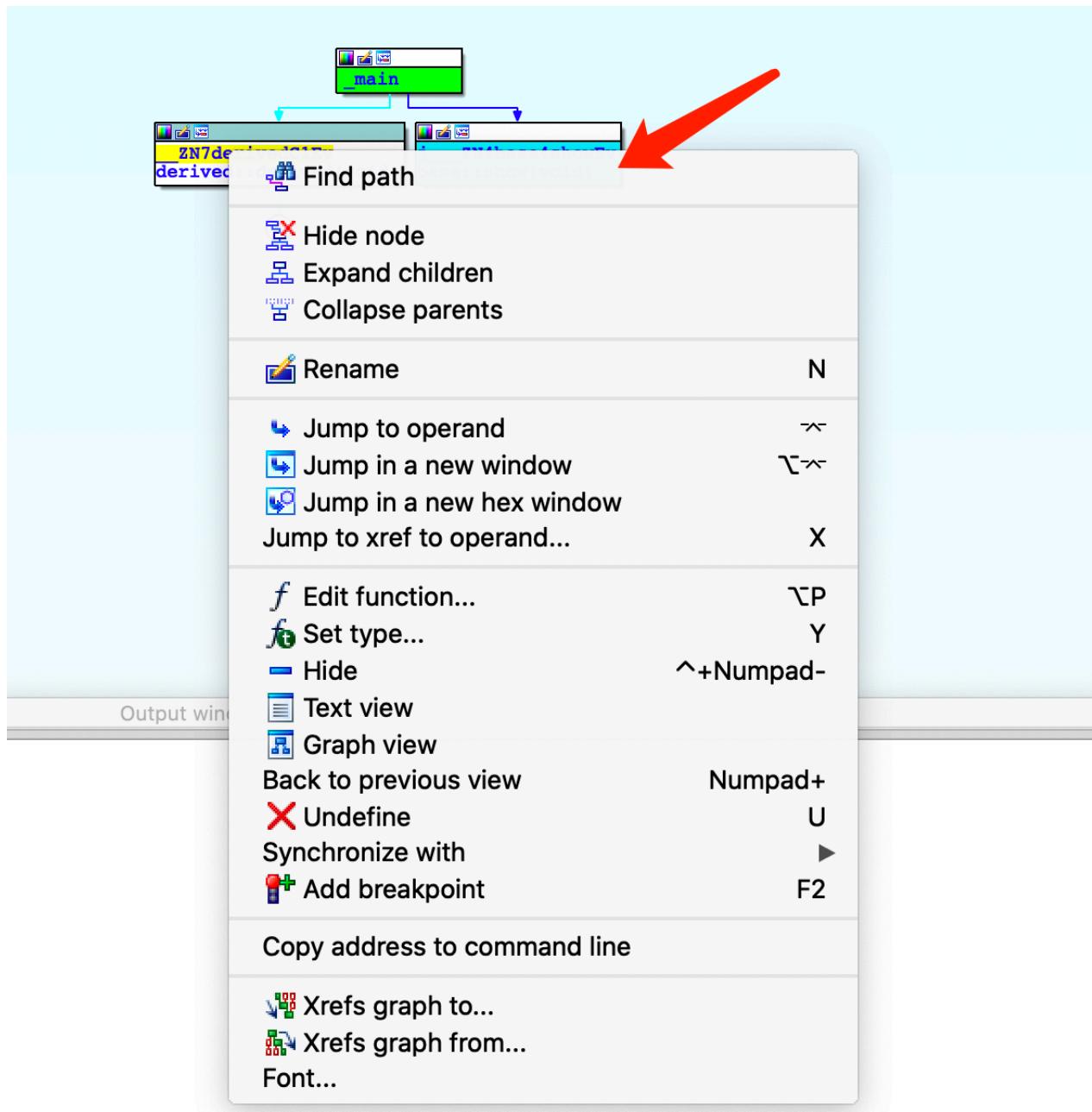
```

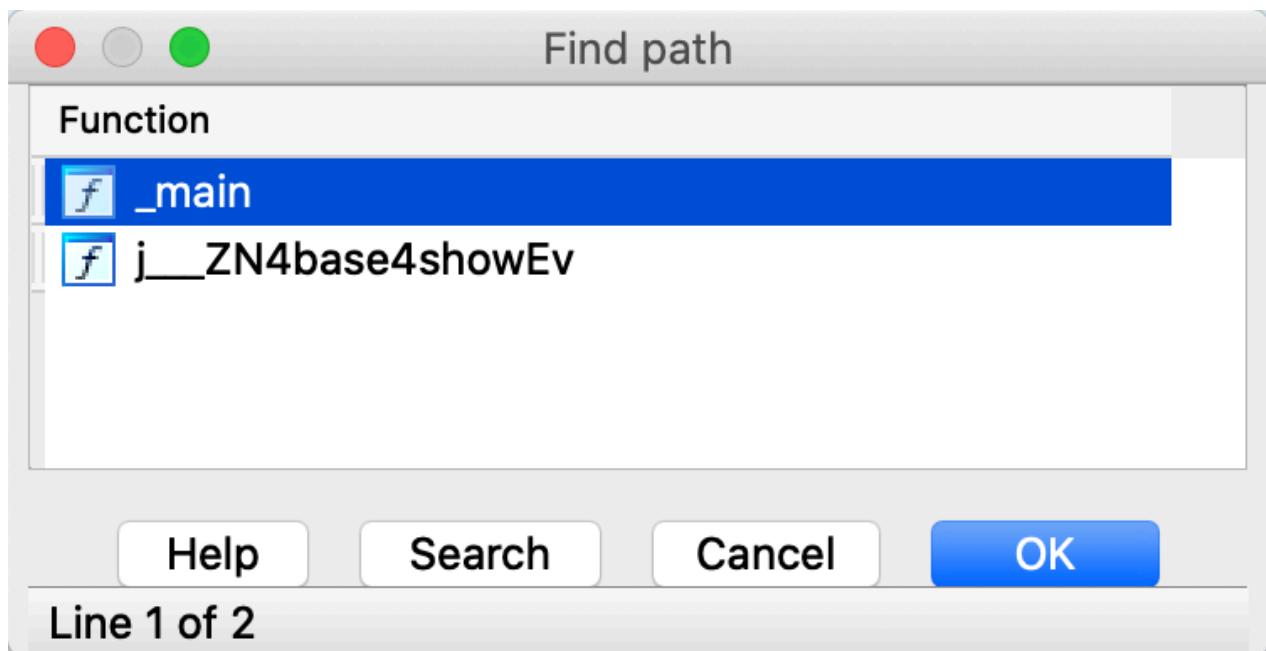
text:0000000100001020 ; ===== S U B R O U T I N E =====
text:0000000100001020 ; Attributes: bp-based frame
text:0000000100001020 ; int __cdecl main(int argc, const char **argv, const char **envp)
text:0000000100001020     public _main
text:0000000100001020     proc near
text:0000000100001020 var_10    = qword ptr -10h
text:0000000100001020 var_8     = qword ptr -8
text:0000000100001020
text:0000000100001020 push   rbp
text:0000000100001020 mov    rbp, rsp
text:0000000100001024 sub    rbp, 10h
text:0000000100001028 lea    rdi, [rbp+var_10] ; this
text:000000010000102C call   _ZN7derivedC1Ev ; CODE XREF: main+10
text:0000000100001031 lea    rax, [rbp+var_8]
text:0000000100001035 mov    [rbp+var_8], rax
text:0000000100001039 mov    rax, [rbp+var_8]
text:0000000100001043 mov    rdi, rax
text:0000000100001043 xrefs to derived::derived(void)

text:000000010000104C p _main+C     call  _ZN7derivedC1Ev, derived::derived(void)
text:000000010000104F
text:0000000100001052
text:0000000100001056
text:000000010000105B
text:0000000100001061
text:0000000100001062
text:0000000100001062 main
text:0000000100001062
text:0000000100001062 ; ---
text:0000000100001070 Line 1 of 1
text:0000000100001070 ; ===== Line 1 of 1 =====
text:0000000100001070 ; Attributes: bp-based frame
text:0000000100001070 j int64  fastcall derived::derived(derived * __hidden this)
text:0000000100001070     _ZN7derivedC1Ev proc near ; CODE XREF: main+Ctp
text:0000000100001070
text:0000000100001070 var_8    = qword ptr -8
text:0000000100001070 push   rbp
text:0000000100001070 mov    rbp, rsp
text:0000000100001074 sub    rbp, 10h
text:0000000100001078 mov    [rbp+var_8], rdi

```

也可以用find path去寻找:





Reverse Engineering IDA Pro – How to do Binary Diffing - Patch Analysis

介绍了bindiff这个工具，写两个代码测试一下

```
#include "stdio.h"

int main(){
    for(int i=0;i<10;i++){
        if (i == 5){
            printf("is there! \n");
            return 0;
        }
        printf("Hello there ! \n");
    }
    return 0;
}
```

```
#include "stdio.h"

int main(){
    for(int i=0;i<10;i++){
        printf("Hello there ! \n");
    }
    return 0;
}
```

找到了相关的文章：<https://www.cnblogs.com/lsdb/p/10543411.html> 自行看一下，这个软件不知道为什么出问题了，什么版本都不好使打不开 就很奇怪，问了点大师傅说可能是书出了问题，让我换用diaphora

Reverse Engineering IDA Pro- Code Coverage Measurement with Dynamo Rio & Lighthouse

这节介绍了两个插件，一个是Dynamo Rio 一个是 Lighthouse代码覆盖率浏览器

<https://github.com/DynamoRIO/dynamorio/wiki>

<https://github.com/gaasedelen/lighthouse>

感觉github上的介绍比视频里的好很多，（但是我没看懂这两个插件有什么好处和帮助）