目录

回顾

1. 程序的特征一般有哪些？PEID 识别程序是什么语言的原理是什么？
   ① OEP 特征
   ② 区段名称
   ③ 链接器版本
   ④ 文件头 magic

PEID 识别的是 OEP 特征

55 8B EC 6A FF 68 ?? ?? ?? ?? 68 ?? ?? ?? ?? 64 A1 00 00 00 00 50 64 89 25 00 00 00 00 83 EC 58

64 A1 00 00 00 00 50 64 89 25 00 00 00 00 83 EC 58

2. IDA 的快捷键有哪些？

G 打开跳转窗口
ESC 返回到上一步
N 重命名
R 将数值转为 ASCI 码字符
回车 进入选中地址所在处
D 定义和转换数据类型
Ctrl+X 交叉引用
; 重复注释
F2 设置断点
F5 将反汇编转为 C 代码

空格　切换图形视图和代码视图
H　转换为 16 进制
C　转换为代码
A　转换为 asci 码字符串
S
B　转换为二进制
F7　单步步入
F8　单步步过
F4　运行到光标处
Q　转换为 16 进制
Ctrl+回车　下一步(与 ESC 相反)
Alt+Q　转换结构体(堆栈窗口)
M　转换枚举类型常量
Ins　创建结构体(结构体)
Alt+T　搜索文本
Shift+F9　打开结构体窗口
Shift+F12　打开字符串窗口
Shift+F3　打开函数窗口
CtrL+F7　运行到函数返回
Ctrl+K　打开局部堆栈窗口
Alt+K　打开设置堆栈窗口
Alt+F4　关闭文件
Ctrl+G　打开跳转指定区段窗口
Ctrl+E　打开选择入口点窗口
P　创建函数
V　改寄存器名称
Shift+;　注释
TAB　将反汇编转为 C 代码（与 F5 一样）
F1　打开帮助文档
?　打开计算器
Ctrl+M　打开标签
Ctrl+W　保存
Alt+P　编辑函数
Ctrl+S　跳转到指定的段
Shift+F5　打开签名窗口
K　修改堆栈变量
Y　设置参数类型
E　设置函数结束
Ctrl+Ins　复制
Alt+X　退出
Alt+F10　创建汇编文件
U　将数据或代码转换为未定义
T　转换结构体偏移

VC 程序内存和编译的一些特征

1. 内存值特点

FDFDFDFD 堆标记
CCCCCCCC 未初始化状态
CDCDCDCD 堆申请空间未初始化状态 new -> malloc -> HeapAlloc -》RtlAllocHeap
40300000 浮点数(IEEE 浮点表示法)
3D9B5DC1 Hash 值(算法不可逆，一般采用方式爆破)
00380000 VirtualAlloc 申请的内存地址(页对齐)
FEEEFEEE 堆释放之后的填充信息
0012FF8C 堆栈地址(5,6 位)

字符串：hello
ASCII 码：68 65 6c 6c 6f
HASH 值：定长，是原数据的摘要信息，不可逆
HASH 算法：68+65+6c+6c+6f = xxxx
典型 Hash 算法：MD5，SHA-1, SHA-256, CRC32。。。。

2. C++编译的一些特点

① 局部变量

- 当函数开头有

```
Push ebp
Mov ebp,esp
```

局部变量是 ebp – xxxx

当函数开头没有 push ebp 之类的代码,只有 SUB ESP,yyy
局部变量是 ESP+xxx ，需要根据前后堆栈的变化来确定
一般来说 xxx < yyy

② 参数

- 当函数开头有

```
Push ebp
Mov ebp,esp
Sub ebp, xxx
```

参数是 ebp + 8 为第一个参数，然后递增

- 当函数开头没有 push ebp 之类的代码，只有 SUB ESP,yyy

参数是 ESP+yyy+4 ，需要根据前后堆栈的变化来确定

③ 调用约定

成员函数、构造函数、析构函数都有 this 指针，所以是 thiscall = this 指针+stdcall
静态成员函数、友元函数没有 this 指针是 stdcall

④ 常用寄存器规则

Ecx 常用于表示 this 指针

Eax 常用于表示返回值

C++ 构造函数

特点 1： ECX 传参

特点 2： CALL 内部 有对 ECX 指向缓冲区进行赋值或者初始化

特点 3： **返回值 EAX 是 this 指针**

C++ 成员函数

特点 1： ECX 传参

特点 2： CALL 内部 有对 ECX 指向缓冲区进行访问或是赋值

C++ 析构函数

特点 1： ECX 传参

特点 2： CALL 内部 有对 ECX 指向缓冲区进行访问或是释放

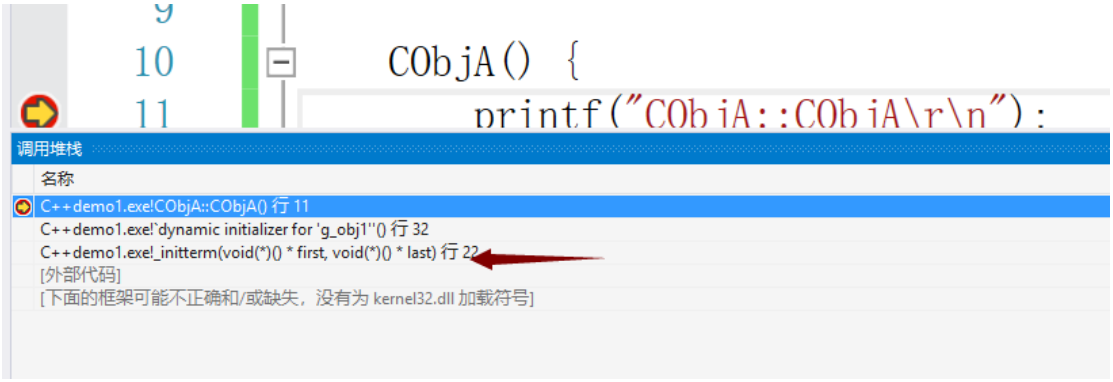
特点 3： 局部对象一般调用是在函数退出前

C++ 全局对象的构造

在 C 运行库中的_initterm 函数中进行的调用

```
218             if (_initterm_e(__xi_a, __xi_z) != 0)
219                 return 255;
220
221             _initterm(__xc_a, __xc_z);  已用时间 <= 1ms
222
223             __scrt_current_native_startup_state = __sc
```

全局对象的构造函数执行的地方



源码：initterm.cpp 中的_initterm 函数

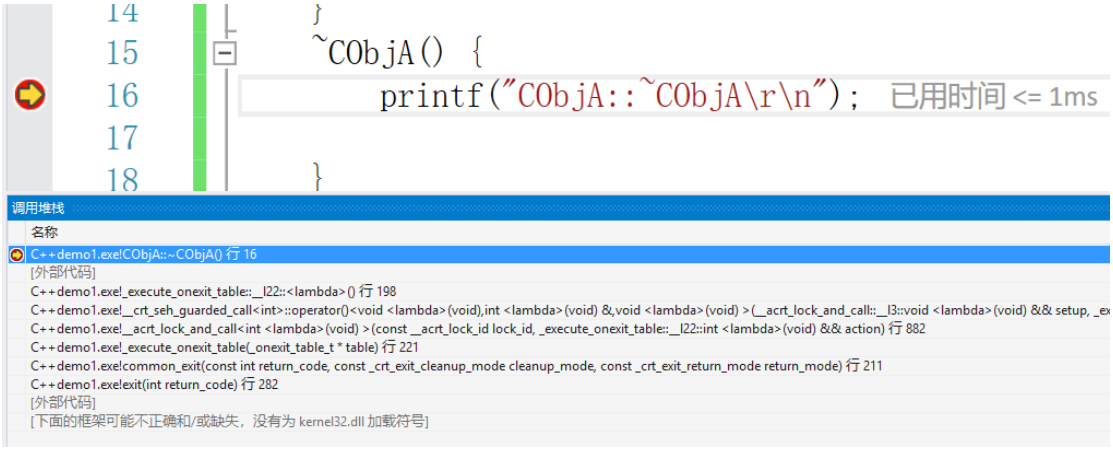
```
extern "C" void __cdecl _initterm(_PVFV* const first, _PVFV* const last)
{
    for (_PVFV* it = first; it != last; ++it)
    {
        if (*it == nullptr)
            continue;

        (**it)(); // 循环调用全局对象的构造函数。
    }
}
```

反汇编信息：

```
        (**it)();
0133A215 8B 4D F8              mov         ecx,dword ptr [ebp-8]
0133A218 8B 11                 mov         edx,dword ptr [ecx]
0133A21A 89 55 F4              mov         dword ptr [ebp-0Ch],edx
0133A21D 8B 4D F4              mov         ecx,dword ptr [ebp-0Ch]
0133A220 FF 15 00 80 3A 01     call        dword ptr [__guard_check_icall_fptr
(013A8000h)]
0133A226 FF 55 F4              call        dword ptr [ebp-0Ch]
```

C++ 全局对象的析构

在 main 函数调用之后，调用的 exit 函数内部



源码：onexit.cpp 中_execute_onexit_table 函数中

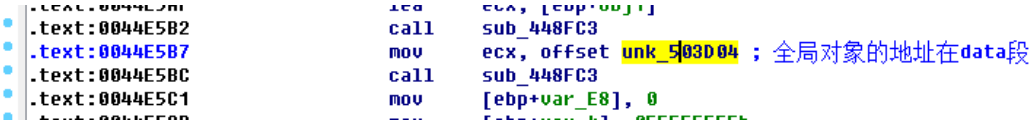
```
            _PVFV const function = __crt_fast_decode_pointer(*last);
            *last = encoded_nullptr;

            function(); // 调用全局对象的析构

            _PVFV* const new_first = __crt_fast_decode_pointer(table->_first);
            _PVFV* const new_last  = __crt_fast_decode_pointer(table->_last);
```

反汇编：

```
            function();
0133758C 8B 45 CC             mov         eax,dword ptr [ebp-34h]
0133758F 89 45 DC             mov         dword ptr [ebp-24h],eax
01337592 8B 4D DC             mov         ecx,dword ptr [ebp-24h]
01337595 FF 15 00 80 3A 01    call        dword ptr [__guard_check_icall_fptr
(013A8000h)]
0133759B FF 55 DC             call        dword ptr [ebp-24h]
```

C++ 全局对象的成员函数

反汇编中，调用全局对象的成员函数时，全局对象的地址会直接赋给 ECX



VS2015 main 特征

```
012EED38   50              PUSH EAX
012EED39   E8 4DB7FFFF     CALL C++demo1.012EA48B
012EED3E   8B00            MOV EAX,DWORD PTR DS:[EAX]
012EED40   50              PUSH EAX
012EED41   E8 09B7FFFF     CALL C++demo1.012EA44F
012EED46   8B08            MOV ECX,DWORD PTR DS:[EAX]
012EED48   51              PUSH ECX
012EED49   E8 13B9FFFF     CALL C++demo1.012EA661
012EED4E   83C4 0C         ADD ESP,0xC
012EED51   5D              POP EBP
```

50 E8 ???????? 8B 00 50 E8 ???????? 8B 08 51 E8

C++ 虚函数

特点 1：当一个类，有一个以上的虚函数定义时，创建类对象，对象基地址就会产生虚函数表指针。
特点 2：有虚函数的类对象，构造函数中有对虚函数表指针进行初始化

```
.text:0044E38F                pop     ecx             ; ---------------
.text:0044E390                mov     [ebp+this], ecx
.text:0044E393                mov     eax, [ebp+this]
.text:0044E396                mov     dword ptr [eax], offset vTablePtr
.text:0044E39C                push    offset aCobjaCobja ; "CObjA::CObjA\r\n"
.text:0044E3A1                call    sub_44AE90
.text:0044E3A6                add     esp, 4
.text:0044E3A9                mov     eax, [ebp+this]
.text:0044E3AC                mov     dword ptr [eax+4], 1
```

虚函数表(vTable)存储在 rdata 段。这说明虚函数表是在编译时就已经生成了。
VC 编译器为每一个有虚函数的类，都会生成相应的虚函数表

当两个类有继承关系时，子类的构造函数代码会嵌入父类的构造函数调用代码

```
.text:0044E400                mov     [ebp+this], ecx
.text:0044E403                mov     ecx, [ebp+this]
.text:0044E406                call    j_CObjA
.text:0044E40B                mov     eax, [ebp+this]
.text:0044E40E                mov     dword ptr [eax], offset vTablePtr2
.text:0044E414                mov     eax, [ebp+this]
```

C++ 虚函数的调用

通过 . 和 通过 -> 调用虚函数是不一样的

通过 . 调用虚函数 与正常函数一样，没区别
通过 -> 调用虚函数，会访问虚函数表，一般是寄存器调用

```
    CObjA obj1;
00ACA92D 8D 4D E0             lea         ecx,[obj1]
00ACA930 E8 91 E8 FE FF       call        CObjA::CObjA (0AB91C6h)
00ACA935 C7 45 FC 00 00 00 00 mov         dword ptr [ebp-4],0

    obj1.Print();
00ACA93C 8D 4D E0             lea         ecx,[obj1]
00ACA93F E8 3A 08 FF FF       call        CObjA::Print (0ABB17Eh)

    CObjA* p1 = &obj1;
00ACA944 8D 45 E0             lea         eax,[obj1]
00ACA947 89 45 D4             mov         dword ptr [p1],eax
    p1->Print();
00ACA94A 8B 45 D4             mov         eax,dword ptr [p1]
00ACA94D 8B 10                mov         edx,dword ptr [eax]
00ACA94F 8B F4                mov         esi,esp
00ACA951 8B 4D D4             mov         ecx,dword ptr [p1]
00ACA954 8B 02                mov         eax,dword ptr [edx]
00ACA956 FF D0                call        eax
```

```
.text:0045A944    lea     eax, [ebp+objA]
.text:0045A947    mov     [ebp+pobjA], eax
.text:0045A94A    mov     eax, [ebp+pobjA]
.text:0045A94D    mov     edx, [eax]      ; edx = [eax] = [pObjA] = 虚函数表指针
.text:0045A94F    mov     esi, esp
.text:0045A951    mov     ecx, [ebp+pobjA]
.text:0045A954    mov     eax, [edx]      ; eax = [edx] = [虚函数表指针] = 第一个虚函数
.text:0045A956    call    eax
```

C++数据结构

vector, list, map