2021/05/17_x86逆向C++_第4课_虚表、默认构造默认析构的识别

笔记本： x86逆向-C++
创建时间： 2021/5/17 星期一 15:06
作者： ileemi

课前会议

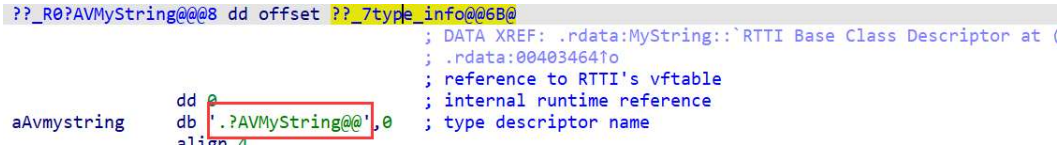
建模工具：
SPA PowerDesigner -- 分析C++类程序，面向对象设计软件，逆向类结构。
struml
Visual Studio -- 类设计器

虚表

如果一个类有虚函数，编译器会为该类生成一个虚表，虚表的位置在一个只读数据区，虚表是一个数组的数据结构。虚表的每一项都指向一个成员函数指针，有几个虚函数，虚表就有几项。虚表不一定以0结尾，虚表的项数只能通过调用或者表的数据特征判定。

> C++类名的识别（高版本VS编译的程序）：类中定义了虚函数，其构造函数和不普通函数的行为不一样，执行构造函数代码前有填写虚表的操作。从虚表首地址往上四个字节，有对应的类名字符串。

```
; const char Source[3]
Source          db 3 dup(0)             ; DATA XREF: sub_401170+22↑o
                dd offset ??_R4MyString@@6B@ ; const MyString::`RTTI Complete Object Locator'
; const MyString::`vftable'
??_7MyString@@6B@ dd offset sub_401060  ; DATA XREF: sub_401010+D↑o
                dd offset ??_R2MyString@@8 ; reference to the array of base classes
; const MyString::`RTTI Complete Object Locator'
??_R4MyString@@6B@ dd 0                 ; DATA XREF: .rdata:00403190↑o
                                        ; signature
                dd 0                    ; offset of this vtable in complete class (from top)
                dd 0                    ; offset of constructor displacement
                dd offset ??_R0?AVMyString@@@8 ; reference to type description
                dd offset ??_R3MyString@@8 ; reference to hierarchy description
; MyString::`RTTI Base Class Array'
??_R2MyString@@8 dd offset ??_R1A@?0A@EA@MyString@@8
```



默认构造函数和析构函数

如果一个类有虚函数，没有构造或者析构函数会生成默认构造函数以及析构函数。

虚表使用的时机

以下特征是识别构造函数和析构函数的一个重要依据：

- 在执行构造函数代码之前会初始化虚表（主要为了填虚表）
- 在执行析构函数代码之前会还原虚表，析构中也需要填写虚表（考虑到继承，以及虚表中调用虚函数）。主要是为了还原虚表防止基类的析构中调用派生类的虚函数。

虚表的识别

存在虚表时，对象首地址存放着虚表的指针 vtptr（在只读数据区）。在构造函数代码执行之前，编译器填虚表项。虚表中的成员是指向类中对应的虚函数地址（函数指针）。虚表项的结尾不一定以0结尾（需要根据汇编的上下文具体分析）。

> 虚函数的调用会走虚表。所以在执行类构造函数代码前填写虚表项。当类中没有构造函数时，编译器会为其生成默认构造（用于填写虚表）。

通过关闭 "运行时类型信息"，通过虚表识别类名的方法就不行，但是C++的大多标准语法会不支持使用（高版本的编译器会强制开启该功能）。

构造函数不具备多态性，所以在构造函数、析构函数中调用虚函数不会查询虚表，直接 call 函数地址。

单重继承的判断

子类需要先调用基类的构造再覆盖自己的虚表。判断类是否有父类，需要看构造函数中填虚表之前是否有代码（调用基类构造的代码）。

- 高版本VS：成员对象的构造不会在填写虚表之前
- VC++6.0：成员对象的构造会在填写虚表之前
  add ecx, 4
  call xxx
  老版本根据this指针判断基类还是成员对象构造。

没有虚函数的继承类再release编译后看不出类的继承层次，构造析构按照类的继承层次依次调用。

组合（类继承中，各个类都没有虚函数）可以取代继承，内存结构相同。

```
// 继承 虚表会覆盖，高版本在填写虚表之前，低版本在填写虚表之后
class Line :public Shape{
};

// 组合 add ecx, 4
class Line {
private:
  Shape shape;
};
```

寻找虚表的方法

- 通过构造函数获取虚表位置
- 通过析构函数获取虚表位置
- 通过调用虚函数的位置获取虚表位置

找间接调用，虚函数的调用必定会通过虚表来进行，识别出虚表，通过虚表来间接调用基本就能找到。
存在虚表时，对象首地址存放着虚表的指针 vtpt。

定位单重继承类的构造、析构和对象

首先找到虚表，查看虚表的引用（引用的项目中，析构只有一项，其余全是构造）