2021/05/14_x86逆向C++_第3课_对象作为函数参数的识别

笔记本: x86逆向-C++
创建时间: 2021/5/15 星期六 10:54
作者: ileemi

还原C++可执行程序时，需要从 `_initterm` 开始还原（可能存储全局类对象）。

工具类中成员函数类型前需要加static。工具类通用，不应该也不会产生对象。

C++工具类的定义以及使用。

对象作为函数参数的识别

> 传递类对象指针，是没有意义的（面向过程）。将对象地址当作函数参数进行传递，不会是thiscall。函数参数传递引用类对象，相当于传递类对象指针（不存在拷贝）。

以类对象当作函数参数进行传递时，会产生拷贝构造（类中没有拷贝对象，默认使用memcpy（浅拷贝），类较小时，编译器不使用memcpy，会自己拷贝）。

```
sub     esp, 0Ch
mov     eax, esp
mov     ecx, [ebp-20h]
mov     [eax], ecx
mov     edx, [ebp-1Ch]
mov     [eax+4], edx
mov     ecx, [ebp-18h]
mov     [eax+8], ecx
call    showPerson
add     esp, 0Ch
```

移动构造函数（左值引用 CTest&&），出现拷贝构造时，会自动释放类对象的this指针，功能和引用计数相似。

结构体对象当作函数参数时，不会传递this指针。构造函数的调用约定默认为thiscall，即便是修改了也没有。

移动构造函数的参数是对象指针。
CPerson(CPerson&& obj); //==> CPerson(CPerson* obj)

对象为参数的传递方式

指针
浅拷贝，出作用域析构
深拷贝，出作用域解析

小对象：直接push
大对象：往栈顶进行拷贝

```
mov edi, esp ;
rep movsd
```

拷贝构造函数

是一个带参数的构造函数。
有拷贝构造：mov ecx, esp，在函数外调用构造函数，函数内部调用析构。
无拷贝构造：抬高栈顶，以栈顶为目标执行memcpy复制原对象。
识别：

```
mov ecx, esp ; 识别标志， 将栈顶当作this指针，大概率使用了拷贝构造。
...
call xxx ; -- 返回this指针
```

ecx 为this指针且指向栈顶，函数内部有直接使用 ecx。

识别特征：参数、函数外进行了拷贝构造，函数内return前调用析构

对象作为函数返回值的识别

指针对象
临时对象：对应的函数中返回时会调用移动构造函数。
返回对象，浅拷贝
返回对象，深拷贝
返回引用对象
无名对象：当作局部对象处理
//CPerson obj2 = CPerson(3); //无名对象作用域在本行代码，等价与 CPerson obj2(3);