空指针调用函数

1- 空指针可以调用非虚函数，只要不进行解引用，就不会发生段错误
2- 早绑定：能够在编译时候搞定的事情绝不拖到运行时搞定

首先看一段代码是否知道其正确还是错误。

```
class A{
public:
    void print()
    {
        cout << "Hello" << endl;
    }

};
void main()
{
    A *a = NULL;
    a.print();
}
```

问你程序是否正确执行，或者执行结果是什么。

- 这就是一个典型的表示C++是一个静态语言的特征。可以阐明“静态绑定”和“动态绑定”的区别。
- 真正的原因是：因为对于非虚成员函数， C++这门语言是静态绑定的。这也是C++语言和其它语言Java, Python的一个显著区别。C++奉行的原则是**能够在编译时候搞定的事情绝不拖到运行时搞定**。
- 我们的第一感觉是这个程序应该是错的，然而事实却是相反的。每个对象都有指向自己的this指针，指针的值会因为不同的对象不同而不同，用来区别不同的对象。这里的this指针就是一个NULL。如果我们调用this指针的时候就会出错。可是这个函数并没有用到this指针就是简单的输出字符串，这一过程在编译阶段就已经完成了。所以不会报错。这一点与JAVA等动态语言不同。
- 但是对于C++。为了保证程序的运行时效率， C++的设计者认为凡是编译时能确定的事情，就不要拖到运行时再查找了。所以C++的编译器看到这句话会这么干：
- 1：查找a的类型，发现它有一个非虚的成员函数叫print。（编译器干的）
- 2：找到了，在这里生成一个函数调用，直接调A::print()。
- 所以到了运行时，由于print函数里面并没有任何需要解引用some null指针的代码，所以真实情况下也不会引发segment fault。这里对成员函数的解析，和查找其对应的代码的工作都是在编译阶段完成而非运行时完成的，这就是所谓的静态绑定，也叫早绑定。

正确理解C++的静态绑定可以理解一些特殊情况下C++的行为。