C++ 空基类优化（EBO）与私有继承

Created on 2020 / 04 / 27, 20:27:20

#Cpp

通常我们认为"私有继承"和"组合"都有着相同的应用层语义，即：**派生类或包含类是通过继承的类或组合的类对象来实现的**（is-implemented-in-terms-of）。但在 C++ 中，由于 EBO（**Empty Base Optimization**）机制的存在，某种情况下我们可能会更倾向于使用私有继承而非组合的方式。

通常来说，C++ 为了保证相同类型的不同对象其在内存中的地址始终是不同的，因此对于一个即使没有任何非静态（non-static）成员的空类，其对象也必须保证大小最小为 1 个字节。如下代码所示，对于空类 A，我们可以看到其对象大小为 1 个字节。

```cpp
class A {};
int main(int argc, char **argv) {
  std::cout << sizeof(A) << std::endl;  // 1;
  return 0;
}
```

在这种情况下，由于类成员数据对齐（内存对齐）的存在，便可能产生：**以组合方式包含的空类A，导致整个类对象的大小有着近翻倍的增长**。比如在下述代码中，按照我们的想法，类 B 对象的大小应该为5字节（空类 1 字节，int 成员变量 4 字节）。但实际情况是，由于数据成员的对齐，B 对象的实际大小为 8 字节，相较增大了 60%。

```cpp
class A {};
class B {
  A vA;
  int x;
};
int main(int argc, char **argv) {
  std::cout << sizeof(B) << std::endl;  // 8;
  return 0;
}
```

然而不同的是，由于 EBO 机制的存在，对于继承了该空类的子对象而言，这部分"多余的内存"占用便会被优化。如下代码所示，当类 B 继承自空类 A 时，B 类对象的大小等于其唯一的 int 整型成员变量的大小。

```cpp
class A {
  using Int = int;
  typedef double Double;
  enum class Config : uint8_t { Timeout = 1 };
  void foo() {}
  static int globalVal;
};
class B : public A { int x; };
int main(int argc, char **argv) {
  std::cout << sizeof(A) << std::endl;  // 1;
  std::cout << std::boolalpha
    << (sizeof(B) == sizeof(int)) << std::endl;  // true;
  return 0;
}
```

这里需要注意的是，本文中我们所说的"空类"，是指不含有 "non-static" 成员变量的类，而对于**类型定义**、**成员函数定义**、**静态成员变量**以及**枚举类**（scoped/unscopeded）均可以出现在空基类中，而不会产生额外的内存占用。另一方面需要强调的是，EBO 是对所有继承的派生类均可见的，不会根据继承访问控制的类型而做出区分。本文标题我们所说的 private 私有继承，是因为 EBO 的一般应用场景为私有继承，即派生类基于基类的 "is-implemented-in-terms-of" 实现方式。

Last built on 2021 / 09 / 25, 15:38:46

⇐ Next Post

Older Post ⇒