链滴

不要定义原型相同但实现不同的同名内联函数

作者：localvar

原文链接：https://ld246.com/article/1521882643357

来源网站：链滴

许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

最早是在下面的程序中发现的问题：

```
// a.cpp
struct foo
{
    void bar(){printf("foo::bar in a.cpp\n");}
};

void testa()
{
  foo f;
  f.bar();
}

// b.cpp
struct foo
{
    void bar(){printf("foo::bar in b.cpp\n");}
};

voidtestb()
{
  foo f;
  f.bar();
}

// main.cpp
int main(int argc, char** argv)
{
  testa();
  testb();
  return 0;
};
```

结果我发现这个程序居然输出:

```
foo::bar in a.cpp
foo::bar in a.cpp
```

先怀疑是编译器的bug, 不过很快否定了(因为编译阶段并不管具体链接的符号). 继而怀疑是链接器的bug, 但最后发现还是我的bug, 因为上面的代码实际上等价于：

```
// test.h
struct foo
{
    inline void bar();
};

// a.cpp
#include "test.h"
inline void foo::bar(){printf("foo::bar in a.cpp\n");}

void testa()
{
```

```
  foo f;
  f.bar();
}

// b.cpp
#include "test.h"
inline void foo::bar(){printf("foo::bar in b.cpp\n");}

void testb()
{
  foo f;
  f.bar();
}

// main.cpp
int main(int argc, char** argv)
{
  testa();
  testb();
  return 0;
};
```

由于foo::bar是内联的, 所以允许每个编译单元有一个自己的实现, 但在链接阶段, 链接器是不可能发现两个实现不同的(理论上可能,但工作量太大了), 所以它会假设实现相同(毕竟大家一般都把内联函数写头文件中), 并把所有的引用指向同一个实现(如果分别指向本编译单元内部自有的实现肯定是错的, 考函数中定义了静态变量的情形), 这也就出现了本文中的问题. 如果foo::bar不是内联的, 则会有一个链错误。

这里bar是作为一个成员函数出现的, 但它是非成员函数也有同样的问题, 所以结论就是: 不要在同一个字空间中定义原型相同但实现不同的同名内联函数。

ps: 狗年最后一篇，祝大家新春愉快，万事如意，合家欢乐。