C++ 反射：从设计到实现

介绍

概念

反射机制允许程序在运行时借助Reflection API取得任何类的内部信息，并能直接操作对象的内部属性和方法

问题

1. C++ 不支持反射
2. 很多业务场景需要依赖反射机制，比如：RPC、WEB MVC、对象序列化等

目标

1. 类对象反射
2. 类成员数据反射
3. 类成员函数反射

演示

类定义：

```
class A : public Object
{
public:
    A() {}
    ~A() {}

    void f1()
    {
        std::cout << "f1" << std::endl;
    }
```

```
11
12      void f2()
13      {
14          std::cout << "f2" << std::endl;
15      }
16
17  public:
18      string m_name;
19      int m_age;
20  };
21
22  REGISTER_CLASS(A);
23  REGISTER_CLASS_FIELD(A, m_name, string);
24  REGISTER_CLASS_FIELD(A, m_age, int);
25  REGISTER_CLASS_METHOD(A, f1, void, void);
26  REGISTER_CLASS_METHOD(A, f2, void, void);
```

三种类型的宏

1、类对象注册宏：

```
1  REGISTER_CLASS(A);
```

2、类成员数据注册宏：

```
1  REGISTER_CLASS_FIELD(A, m_name, string);
```

3、类成员函数注册宏：

```
1  REGISTER_CLASS_METHOD(A, f1, void, void);
```

反射用法：

```cpp
int main()
{
    // 类对象反射
    ClassFactory * factory = Singleton<ClassFactory>::instance();
    Object * a = factory->create_class("A");

    // 类成员数据反射
    string name;
    a->get_field_value("m_name", name);
    a->set_field_value("m_name", "kitty");

    int age;
    a->get_field_value("m_age", age);
    a->set_field_value("m_age", 30);

    // 类成员函数反射
    a->call("f1");
    a->call("f2");

    return 0;
}
```

类对象反射

简单地说就是程序运行时读进来一个字符串，然后程序就会自动创建出字符串对应名字的类对象

```cpp
class A {};
class B {};

void * create_class(const string & className)
{
    if (className == "A")
    {
        return new A();
    }
    else if (className == "B")
    {
        return new B();
    }
    else
    {
        return nullptr;
    }
```

```
18 }
```

类成员数据反射

关键问题：如何获取类成员数据的内存地址偏移量（offset）

黑科技1：

```
1 #define OFFSET(className, fieldName) \
2     ((size_t)(&((className *)0)->fieldName))
```

黑科技2：

```
1 Test t;
2 size_t offset = (size_t)(&(t.member)) - (size_t)(&t);
```

类成员函数反射

类成员函数指针：

```
1 typedef std::function<void(Test *)> test_method;
2 method method = &Test::f1;
3
4 Test * t = new Test();
5 method(t);
6
7 uintptr_t ptr = (uintptr_t)&method;
8 (*(test_method *)(ptr))(t);
```