

66 Type Punning in C++

1. 类型系统

Type punning (类型双关) 只是一个花哨的术语, 用来在C++中绕过类型系统。C++是强类型语言, 也就是说它有一个类型系统, 不像JavaScript那样创建变量不需要声明变量类型, 但C++中你创建变量时必须声明整数、双精度数、结构体等等类型。然而这种类型系统并不像Java中那么“强制”, C++中虽然类型是由编译器强制执行的, 但你可以直接访问内存, 所以可以很容易地绕过类型系统, 你是否要这么做取决于你的实际需求。在某些情况下, 你绝对不应该规避类型系统, 因为类型系统存在是有原因的, 除非你有充分的理由, 否则你不会想过多地使用它。

假设我有一个简单的类, 现在想把它写成一个字节流, 就可以重新解释它的整个结构, 将它作为一个字节数组然后用字节流输出出来。很多情况下这这是非常有用的, 这是一种原始的、底层的访问, 这就是为什么C++效率高, 应用程序性能好的原因了。

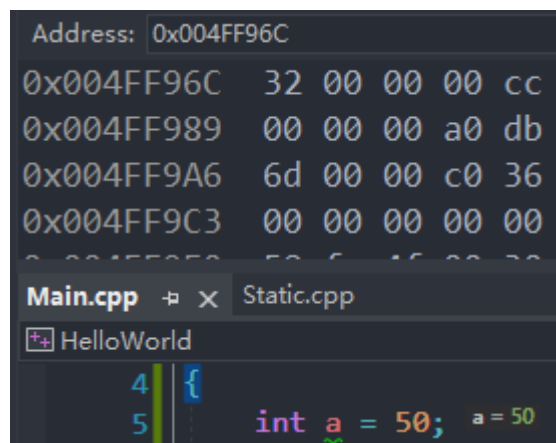
```
#include <iostream>

int main()
{
    int a = 50;
    double value = a;

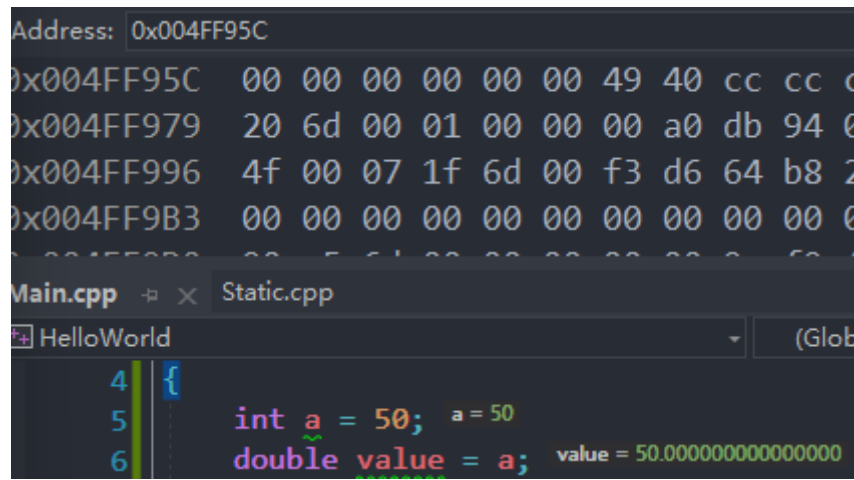
    std::cout << value << std::endl;

    std::cin.get();
}
```

内存中查看a:



内存中查看value:



这个例子中是一个隐式转换，显式转换只需要改为：

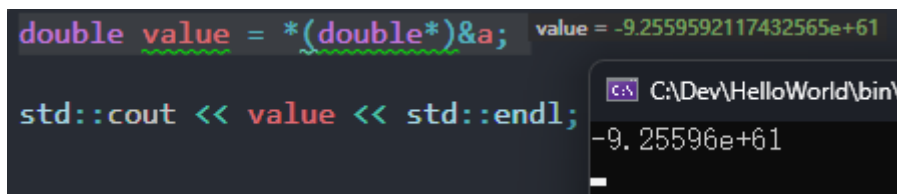
C++

```
double value = (double)a;
```

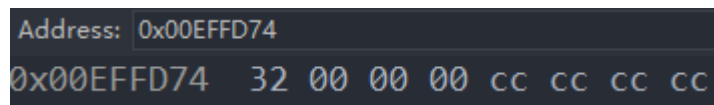
那如何取a的那段内存，让这段地址被当做 **double** 来看待呢？

C++

```
double value = *(double*)&a; // 原始方法：对a取地址，此时变为int指针，将类型修改为double指针后解引用
```



查看value的地址，因为double有8个字节，所以剩下的是未初始化的内存。



导致发生的原因是在上面的类型转换做的很糟糕，因为它们的大小不同，我们取了一个4字节的int，然后定为double。我们在这里做的是先把一个int型指针转换为了一个double型指针，再解引用，它实际上是在我们的int后继续了4个字节然后获取了这部分内存，它并不是我们用来存放a的内存。这很糟糕，在某些情况下甚至会导致崩溃。

这里的意思是我们已经把内存复制到了一个新的double块中操作是安全的，但是读取了不属于我们的内存是不好的。

如果你不想新建一个变量，只是想把这个int当做double来访问，只需要再double后加一个&，引用而不拷贝，这样你就可以编辑int的内存，这是很危险的，因为double需要8个字节而我们的空间只有4个字节，这可能会导致程序崩溃。

1. 结构体类型转换

C++

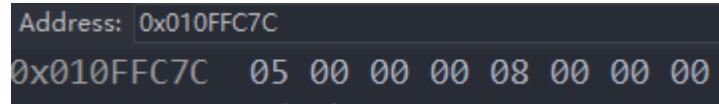
```

struct Entity
{
    int x, y;
};

int main()
{
    Entity e = { 5, 8 };
    std::cin.get();
}

```

在内存中，这个结构体其实就是由2个int组成的，就是这两个整数x,y。



```

Address: 0x010FFC7C
0x010FFC7C  05 00 00 00 08 00 00 00

```

结构体本身不包含任何类型的填充，任何类型的数据，如果是一个空的结构体，那么它至少是1个字节，因为我们需要对这段内存进行寻址，但如果结构体中有变量，比如这个int x和y，那它就只有这两个整数。因此我们可以将Entity结构体看成一个int数组，并且不用e.x, e.y这种方法读取这些整数。

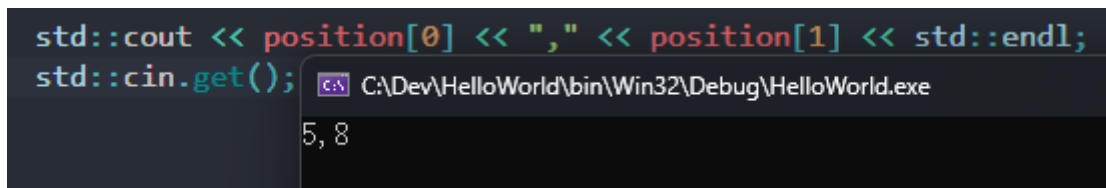
C++

```

int* position = (int*)&e;
std::cout << position[0] << "," << position[1] << std::endl;

```

因为我们将其转换成了数组，所以可以像访问数组那样访问它。



```

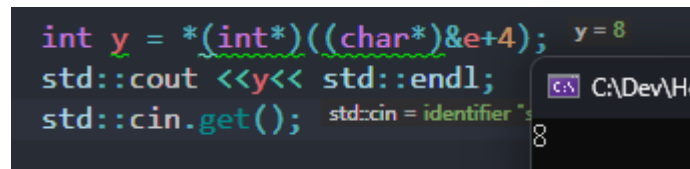
std::cout << position[0] << "," << position[1] << std::endl;
std::cin.get();

```

C:\Dev\HelloWorld\bin\Win32\Debug\HelloWorld.exe

5, 8

同理的更疯狂的操作：



```

int y = *(int*)((char*)&e+4); y = 8
std::cout << y << std::endl;
std::cin.get();

```

C:\Dev\HelloWorld\bin\Win32\Debug\HelloWorld.exe

8

可以看出C++是一种强大的语言的一个重要原因就是它可以自如地操纵内存。