# 21 Static in C++

类的内外（struct,class), 实例化见本篇

| 类外 | 类内 |
| --- | --- |
| 修饰的符号在link阶段是局部的，只对定义它的编译单元(.obj)可见 | 这部分内存是这个类的所有实例共享的，多次实例化，static变量也只会有一个实i（见下篇） |

就算你实例化了很多次这类或者结构体，但那个*static*变量只会有一个实例，类里面的*static method*也是一样，方法里也没有改实例的指针(*this*)

静态变量存储在静态存储区中，而不是栈或堆上。它们在程序的整个执行期间都存在，并且只有一个实例。

static变量或函数表示在*link*到它实际的定义时，*linker*不会在这个编译单元(.obj)外面找到它的定义，有点像类的*private*属性

```C++
\\ Static.cpp
static int s_Variable = 5;  // s stands for static

\\ Main.cpp
#include <iostream>

int s_Variable = 10;

int main()
{
    std::cout << s_Variable << std::endl;    // 10
    std::cin.get();
}
```

```C++
\\ Static.cpp
int s_Variable = 5;  // remove static

\\ Main.cpp
#include <iostream>

int s_Variable = 10;

int main()
{
    std::cout << s_Variable << std::endl;    // LNK ERROR already defined in Main.obj
    std::cin.get();
}
```

所以两个全局变量的名字不能一样

## 解决方案1: extern link

*extern*会在另外的编译单元里找s_variable的定义，*external linkage or external linking*

```C++
\\ Static.cpp
int s_Variable = 5;

\\ Main.cpp
extern int s_Variable;    // 它是变量的引用

std::cout << s_Variable << std::endl;  // 5
```

即你定义的函数和变量只对它的声明所在的cpp文件(编译单元)“可见”
*global*很不好，容易产生bug。