动态库并发调用与主键必要性

# 动态库并发调用

## 结论

多个程序引用同一个动态库，单例仅存在于各自的进程中，不同进程中的单例对象互不影响。

## 测试工程

见SingletonLib测试工程，启动两个TestConsole控制台（引用的是同一个单例类SingletonLib），通过输入“+”、“-”分别进行计数，两个控制台中SingletonLib对象的计算互不干扰。

## DLL调用机制

1. 微软支持

参考资料：微软支持网页

<https://support.microsoft.com/zh-cn/help/815065/what-is-a-dll>

内容：

“通过使用DLL，程序可以实现模块化，由相对独立的组件组成。例如，一个计帐程序可以按模块来销售。可以在运行时将各个模块加载到主程序中（如果安装了相应模块）。因为模块是彼此独立的，所以程序的加载速度更快，而且模块只在相应的功能被请求时才加载。”

1. Windows DLL基本原理与加载连接的实现

<https://blog.csdn.net/zhenyongyuan123/article/details/4063749>

“DLL是一种磁盘文件（通常带有DLL扩展名，是标准win32可执行文件－“PE”格式），它由全局数据、服务函数和资源组成，在运行时被系统加载到进程的虚拟空间中，成为调用进程的一部分，进程中所有线程都可以调用其中的函数。如果与其它DLL之间没有冲突，该文件通常映射到进程虚拟空间的同一地址上。……**在Win32环境中，每个进程都复制了自己的读/写全局变量。如果想要与其它进程共享内存，必须使用内存映射文件或者声明一个共享数据段**。DLL模块需要的堆栈内存都是从运行进程的堆栈中分配出来的。”

从原理上多进程调用DLL彼此互不影响。

# 主键必要性

## 主键必要性

参考资料：Vortex OpenSplice DDS IDL PreProcessor User Manual

内容：第7.1节 Defining Keys

“The identifier <key> is the member of a struct. For a struct either no key list is defined, in which case no specialized interfaces (TypeSupport, DataReader and DataWriter) are generated for the struct, or a key list with or without keys is defined, in which case the specialized interfaces are generated for the struct.”

首先keylist是需要在IDL中定义的，否则IDL编译器不会对主题结构体创建相应的TypeSupport、DataReader、DataWriter。但是keylist后面的key不一定需要定义，如下：

|  |
| --- |
| struct SYNodeDetect  {  string node\_name;  long msg\_type;  long node\_type;  };  #pragma keylist SYNodeDetect |

SYNodeDetect后可以不设置具体的key。这种情况下IDL编译器依然会对主题结构体创建相应的TypeSupport、DataReader、DataWriter，与此同时也可避免所有数据都要定义主键的问题。对于没有主键的数据，样本之间

目前我们公司的仿真引擎采用的就是上面的方式来定义keylist，详见SimData.idl文件。

## 数据持久性和正确性

（待补充，QoS策略测试中）