# 初始化

在C语言领域，变量初始化的顺序通常不是很重要，链接器只需要在数据段(.data节)中保留存储空间，并将初始值写入该位置即可。变量的初始化其实在程序启动的时候就完成了。

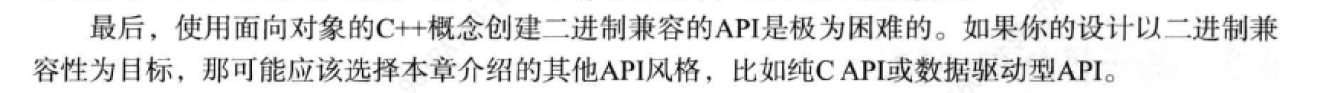
但是在C++中，数据类型通常是一个对象，对象的初始化是在运行时通过对象构造函数的处理完成的，当类构造函数结束执行的时候，对象初始化也就完成了。

# ABI接口

如果动态库的内部功能是通过C++实现的，那么接口设计需要使用类工厂（c++）或者模块（c）的原则来设计接口。

模块原则典型应用是句柄，句柄通常被实现为void类型的指针，（void类型指针也是c++ this指针的前身）。句柄一般通过open函数创建并返回给调用者，在调用其他模块 接口方法时，句柄一般是第一个入参。在无法使用c++的场合，设计c模块完全等同于面向对象中的类工厂。

类工厂这里有歧义，在C++ API设计这本书中是没有提到这种方法。



# 只对外提供必要的符号

这样可以提高库的加载速度，还可以减少同名符号冲突

Linux下默认都是可见的，windows默认都不可见。

Linux下处理方式1+2：

1. makefile中gcc编译选项增加-fvisibility=hidden -fvisibility-inlines-hidden 确保所有符号不可见
2. 函数前面增加修饰符，\_attribute\_ ((visibility(“default”))) 使该函数可见

*注：-fvisibility-inlines-hidden*

*默认隐藏所有内联函数，从而减小导出符号表的大小，既能缩减文件的大 小，还能提高运行性能，我们强烈建议你在编译任何共享库的时候使用该选项。参见 -fvisibility=hidden 选项。*

检查符号时使用nm命令，

nm –D \*.so 可以看到改名字后的各个函数。

# 重复符号处理

## 静态库重复报错

当链接器将目标文件或者两个静态库同时链接到最终的二进制文件中时，如果两个静态库有重复的符号，将会链接失败。如果只是将函数调用注释掉的话不能解决链接错误。因为，链接器首先尝试将来自所有静态库和目标文件中的所有的元素都组合起来。在链接的早期，一旦出现了重复符号，无论是否引用了这些符号，链接器都会报错。Static修饰的可以限定在一个文件中，所有的重复符号都设置成static就可以不报错。

## 动态库重复不报错

将两个动态库链接到二进制应用程序的时候会比静态库宽松，会接受两个动态库中的重复符号，链接不会报错，但是执行的时候会选择一个进行链接，功能出现非预期。

## 解决方案

增加命名空间或者前缀。

如果使用动态加载动态库（dlopen()或者LoadLibrary()）并不会出现重复符号的问题。

## **符号的优先级顺序**：

二进制或者静态库 >动态库导出符号 >其他不导出符号

## LINUX平台下几种情况：

* 1. 当静态库或者可执行文件与动态库重复符号时，动态库的符号优先级低，会被忽略，以前者为准。
  2. 当两个动态库重复符号时，会根据链接顺序（一般是选择靠前的库）选择一个作为唯一的符号代表，即便某个动态库内部又调用了重复符号，也不会用这个库本身的符号，而以选定的唯一符号代表为准。注意：改变函数调用顺序是没用的，因为是发生在链接器的链接阶段，而不是装载器阶段。（本书9.4.2节 147页）
  3. 当静态库、动态库、应用程序中都有同名符号时，如果都使用了static限定符，那么不会出现这种重复符号报错的问题。
  4. 有一种场景，如果有个日志库，被两个动态库分别链接，有个应用程序再链接这两个动态库，而应用程序本身也会链接这个日志库，那么最好是这个日志库是动态库，并设计成单例模式，那么最终的程序中只会有一份实例。如果日志库是静态库的话，两个动态库使用一个日志实例，应用程序还会有个日志实例，所以比较混乱，最好日志库就设计成动态库。（本书156-161页）