# C/C++

1. **在main执行之前和执行之后的代码可能是什么？**

**main函数执行之前**，主要就是初始化系统相关资源：

* 设置栈指针（操作系统开辟栈空间，因此要使用栈指针来访问）
* 初始化静态static变量和global全局变量，即.data段的内容
* 将未初始化部分的全局变量赋初值：数值型short，int，long等为0，bool为FALSE，指针为NULL等等，即.bss段的内容
* 全局对象初始化，在main之前调用构造函数，这是可能会执行前的一些代码
* 将main函数的参数argc，argv等传递给main函数，然后才真正运行main函数
* \_\_attribute\_\_((constructor))

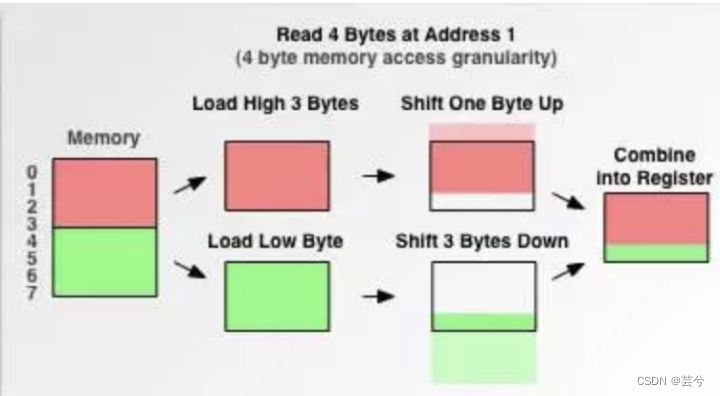
**main函数执行之后：**

* 全局对象的析构函数会在main函数之后执行
* 可以用 atexit 注册一个函数，它会在main 之后执行
* \_\_attribute\_\_((destructor))

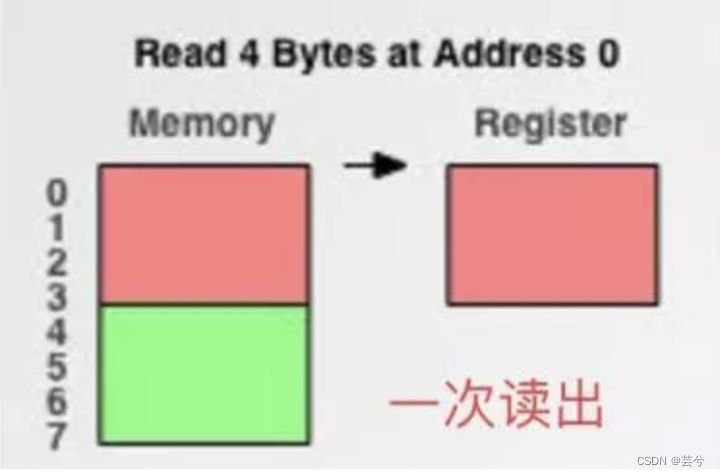
1. **结构体的内存对齐问题？**

* 结构体内成员按照声明顺序存储，第一个成员地址和整个结构体地址相同
* 未特殊说明时，按结构体中size最大的成员对齐（若有double成员，按8字节对齐。）

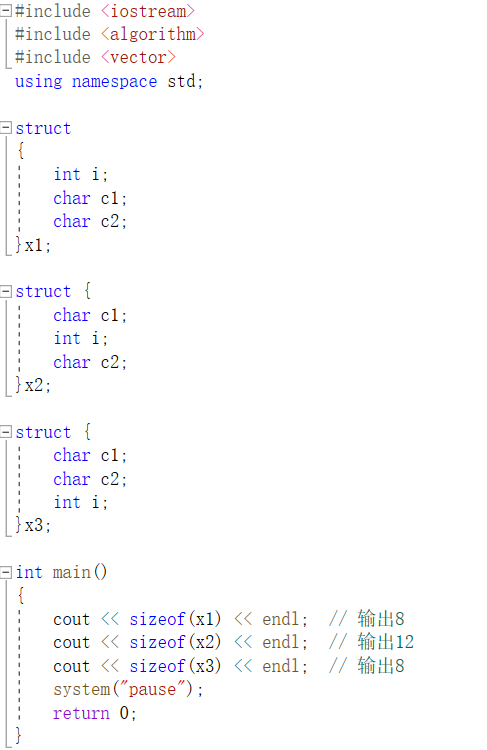
内存对齐的意义：因此假如没有内存对齐机制，数据任意存放，现在一个int变量，存放在地址1开始的连续的4个字节中，当处理器读取时，先从0位置一次读取4字节，提出前边不要的一个字节，再从位置4开始读取4字节块，剔除后边不要的3字节。这个过程很低效。



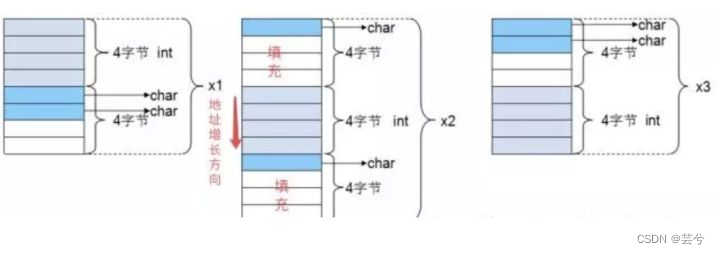
而有了内存对齐，int类型只能根据内存对齐规则存放在自身大小的倍数的位置，就可以一次读出：



下面是一个内存对齐的例子：



具体结果可以参看下面的示意图：



自己总结：

1. **什么是线程同步？**

每一个服务进程的运行，都包含若干进程（Thread），**线程是调度的基本单位，进程则是资源拥有的基本单位**。

线程有自己的私有数据，比如栈和寄存器，同时与其它线程共享相同的虚拟内存和全局变量等资源，当多个线程同时读写同一份**共享资源**的时候，会引起冲突，这时候就需要引入线程同步机制使各个线程排队一个一个的对共享资源进行操作，而不是同时进行。

线程同步用于解决线程访问顺序引发的问题。需要同步的情况可以从如下两方面考虑。

* 同时访问同一内存空间时发生的情况
* 需要指定访问同一内存空间的线程**执行顺序**的情况

1. **半同步/半异步模式？**

首先，半同步/半异步模式中的“同步”和“异步”与前面讨论的IO模型中的“同步”和“异步”是完全不同的概念。在IO模型中，“同步”和“异步”区分的是内核向应用程序通知的是何种IO事件（是就绪事件还是完成事件），以及该由谁来完成IO读写（是应用程序还是内核）。在**并发模式**中，“同步”指的是程序完全按照代码序列的顺序执行；“异步”指的是程序的执行需要由系统事件来驱动。常见的系统事件包括中断、信号等。

半同步/半异步模式中，同步线程用于处理客户逻辑，异步线程用于处理I/O事件。异步线程监听到客户请求后，就将其封装成请求对象并插入请求队列中。请求队列将通知某个工作在同步模式的工作线程来读取并处理该请求对象。

