本章涵盖

ϒ C++ 内存模型的详细信息

ϒ C++ 提供的原子类型

ϒ 标准库

ϒ 这些类型上可用的操作

ϒ 如何使用这些操作来提供线程之间的同步

有一个大多数程序员不会注意到的原因：如果您使用互斥锁来保护数据和条件变量、期货、锁存器或信号事件屏障，那么它们工作的细节并不重要。 只有当你开始尝试“接近机器”时，内存模型的精确细节才变得重要。原子类型和操作正是允许这样做，为低级同步操作提供设施，这些操作通常会减少到一两个 CPU 指令。

5.1 内存模型基础知识

。在 C++ 中，**对象** 并不是面向对象编程中通常所说的“对象”（如类实例），而是指 **内存中的一块存储区域**。

**3. 内存位置（Memory Location）**

* **定义**：
  + 内存位置是存储对象的最小单元。
  + 每个内存位置可以存储一个标量类型（如 int、float、指针等）或一组相邻的位域（bit fields）。
* **标量类型**：
  + 标量类型包括基本类型（如 int、float）和指针类型（如 my\_class\*）。
* **位域（Bit Fields）**：
  + 位域是结构体或类中的特殊成员，用于节省内存。
  + 相邻的位域被视为同一个内存位置，即使它们是不同的对象

你的理解是正确的！**内存位置（memory location）** 是内存的最小单元的一种抽象描述，它表示程序中可以独立寻址和操作的一块存储区域。每个内存位置的大小和具体存储的内容取决于它所存储的 **类型**。

* 例如，如果有三个线程访问同一个内存位置，必须确保每对访问都有一个明确的顺序。

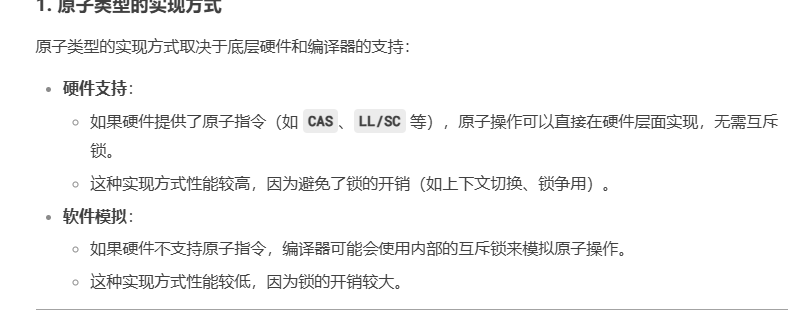
原子操作是不可分割的操作。 你无法从系统中的任何线程观察到这样一个半完成的操作； 它要么完成，要么未完成。 如果读取对象值的加载操作是原子的，并且对该对象的所有修改也是原子的，则该加载将检索对象的初始值或修改之一存储的值。

**1. 原子操作是什么？**

* **定义**：
  + 原子操作是不可分割的操作，要么完全执行，要么完全不执行。
  + 从系统的任何线程中，你都无法观察到原子操作执行到一半的状态。
* **特点**：
  + **不可分割性**：原子操作不会被其他线程中断。
  + **可见性**：原子操作的结果对所有线程立即可见。

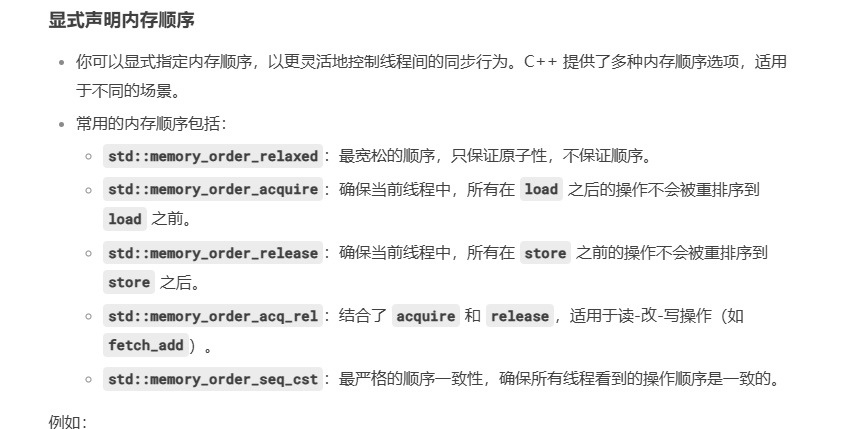
**3. 何时使用原子操作？**

* **适合场景**：
  + 简单的操作（如计数器增减、标志位设置）。
  + 需要高性能的场景（如无锁数据结构）。
* **不适合场景**：
  + 复杂的同步逻辑（如需要保护多个变量的操作）。
  + 在某些平台上，原子操作的性能可能不如预期。



图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成



只有std::memory\_order\_seq\_cst保证所有的线程看见的线程顺序和操作顺序一致，剩下的不同线程看见的线程顺序可能不一致。acquire、release、acq\_rel这三个分别保证了线程内部的一些操作的顺序，relaxed的线程内部的操作顺序也不保证一致对吗





**4. 三者的关系**

* **原子性** 是基础，确保了操作的完整性。
* **顺序性** 和 **可见性** 是多线程编程中更高级的概念，它们确保了操作的正确性和一致性。
* 不同的内存顺序（如 relaxed、acquire、release、seq\_cst）对原子性、顺序性和可见性的保证不同：
  + **std::memory\_order\_relaxed**：只保证原子性，不保证顺序性和可见性。
  + **std::memory\_order\_acquire** 和 **std::memory\_order\_release**：保证原子性和部分顺序性，但不保证全局顺序一致性。
  + **std::memory\_order\_seq\_cst**：保证原子性、顺序性和可见性。

强烈建议避免使用松散的原子操作，除非它们是绝对必要的，即使如此，也要极其谨慎地使用它们。

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

文本

描述已自动生成

文本, 信件

描述已自动生成

文本, 信件

描述已自动生成

在本章中，我已经介绍了C++内存模型的底层细节，以及为线程间同步提供基础的原子操作。这包括由std：：atomic<>类模板的专门化提供的基本原子类型，以及由主std：：atomic<>模板和std：：experimental：：atomic\_shared\_ptr<>模板提供的通用原子接口，对这些类型的操作，以及各种内存排序选项的复杂细节。

我们还研究了栅栏以及如何将它们与原子类型上的操作配对以强制排序。最后，我们回到开头，看看如何使用原子操作来强制不同线程上的非原子操作之间的排序，以及高级设施提供的同步关系。