Memory Model

Learn about multithreading in the C++ Standard Library and its memory model.

WE'LL COVER THE FOLLOWING

Multithreading in C++11

For the first time with C++11, C++ supports native multithreading. This support consists of two parts: A well-defined memory model and a standardized threading interface. We will briefly cover multithreading in the next couple of lessons. However, if you want to study it in detail, take a look at this course Modern C++ Concurrency in Practice: Get the most out of any machine.

Memory Model

The foundation of multithreading is a well-defined memory model. This memory model has to deal with the following points:

- Atomic operations: Operations that can be performed without interruption.
- Partial ordering of operations: Sequence of operations that must not be reordered.
- Visible effects of operations: Guarantees when operations on shared variables are visible in other threads.

The C++ memory model has a lot in common with its predecessor: the Java, memory model. On the contrary, C++ permits the breaking of sequential consistency. The sequential consistency is the default behavior of atomic operations. The sequential consistency provides two guarantees.

1. The instructions of a program are executed in source code order.

2. There is a global order of all operations on all threads.