Introduction to Shared Data

This lesson gives an introduction to how data can be shared between threads in C++.

WE'LL COVER THE FOLLOWING

- Boss-worker model

We only need to think about synchronization if we have shared, mutable data because such data is prone to data races. If we have concurrent non-synchronized read and write access to data, our program will have undefined behavior. The easiest way to visualize concurrent, unsynchronized read and write operations is to write something to `std::cout`. Let's have a look:

```
// coutUnsynchronised.cpp

#include <chrono>
#include <iostream>
#include <thread>

class Worker{
public:
  Worker(std::string n):name(n){};
    void operator() (){
      for (int i = 1; i <= 3; ++i){
        // begin work
        std::this_thread::sleep_for(std::chrono::milliseconds(200));
        // end work
        std::cout << name << ": " << "Work " << i << " done !!!" << std::endl;
      }
    }
private:
  std::string name;
};


int main(){

  std::cout << std::endl;

  std::cout << "Boss: Let's start working.\n\n";

  std::thread herb= std::thread(Worker("Herb"));
  std::thread andrei= std::thread(Worker("  Andrei"));
  std::thread scott= std::thread(Worker("    Scott"));
  std::thread scott= std::thread(Worker("     Scott"));
  std::thread bjarne= std::thread(Worker("      Bjarne"));
  std::thread bart= std::thread(Worker("        Bart"));
  std::thread jenne= std::thread(Worker("          Jenne"));


  herb.join();
  andrei.join();
  scott.join();
  bjarne.join();
  bart.join();
  jenne.join();

  std::cout << "\n" << "Boss: Let's go home." << std::endl;

  std::cout << std::endl;

}
```

Boss-worker model

In the boss-worker model, a single thread - the boss - accepts input for the entire program. Based on that input, the boss passes off specific tasks to one or more worker threads. In the code above, the boss has six workers (lines 29 - 34). Each worker has to take care of 3 work packages. The work package takes 1/5 second (line 13). After the worker finishes the work package, it screams out loudly to the boss (line 15). Once the boss receives these notifications from all workers, it sends them home (line 44).

What a mess for such a simple workflow! The most straightforward solution is to use a mutex, which we will see in the next lesson.