NOTE: this document is incomplete

- a full section covering C++ concurrency, pthreads, and parallel programming/algorithms, and multithreaded programming/algorithms will be added in time

mutexes and locks

- a mutex can be locked directly using try_lock and unlock (Lockable concept)
- std::lock is a function taking a variable number of Lockable arguments and locking them using a deadlock avoidance algorithm
- lock_guard is a scoped lock (designed to unlock at the end of the scope it was instantiated in)
- shared_lock is designed to persist across multiple scopes and be accessed by many threads
- unique_lock is designed to persist across multiple scopes and only be accessed by one thread

Managing threads

- use a thread_guard (constructed with a thread and checks if the thread is joinable on destruction and calls join if so) to make sure a thread is waited on in a scope (listing 2.3)
- call detach to run a thread in the background
- pass arguments to a thread function in the constructor of the thread

Thread pools

- can call std::thread::hardware_concurrency() to get the number of cores
- NOTE: this is not necessarily the optimal number of threads due to hyperthreading for many intel processors hyperthreading may make 2 threads per core optimal but that can increase use `sysctl hw` on mac OS and `lscpu` on linux to find info (logicalcpu max)
- NOTE: use -std=c++11 when compiling manually