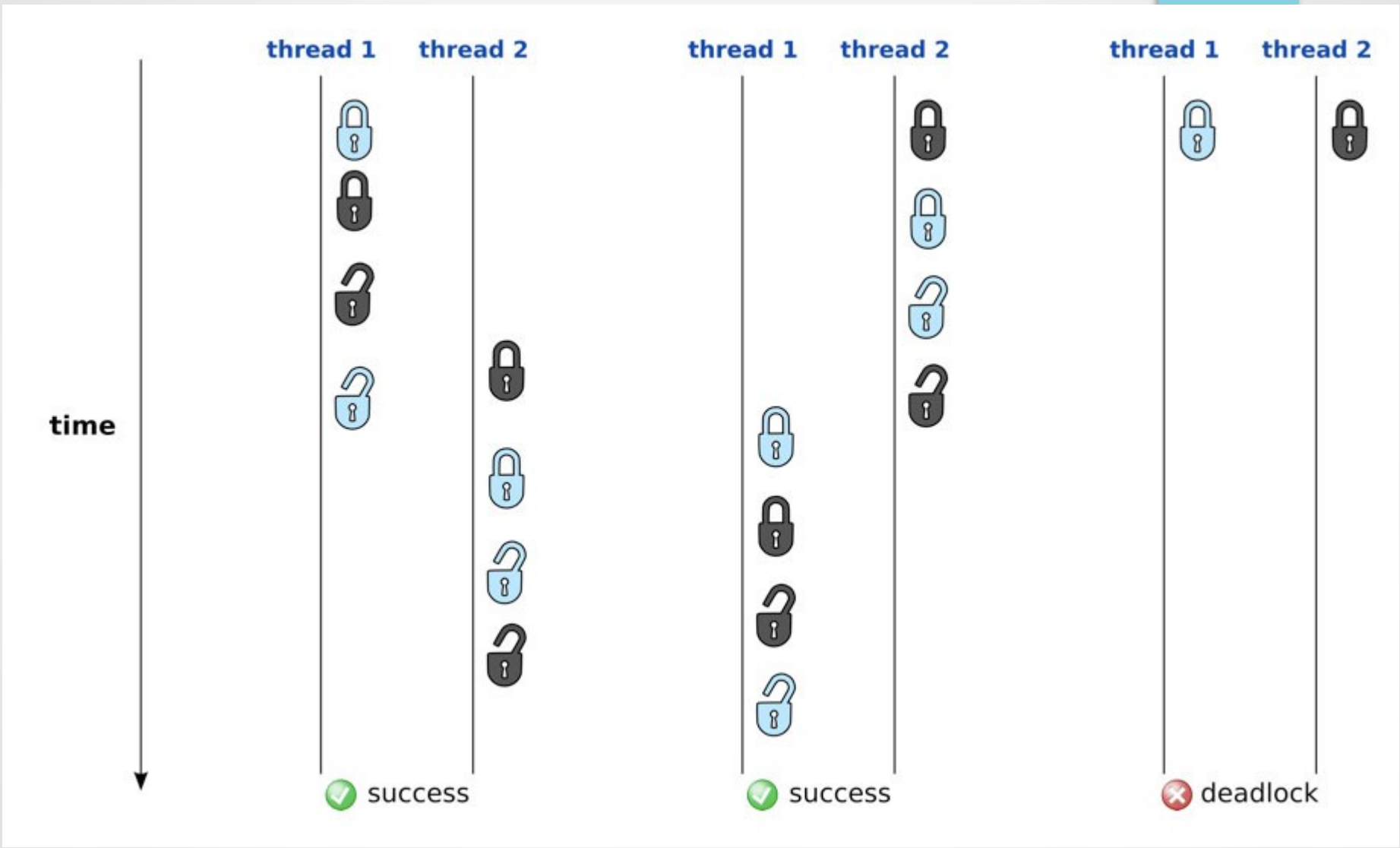
C++ Threads, Atomic & Mutex



Krishna Kumar

C++11 (Threads)

- Perhaps one of the biggest change to the language is the addition of multithreading support. Before C++11 – OpenMP, MPI to target multicore systems.
- **std::threads**
  - The class thread represents a single thread of execution. Threads allow multiple pieces of code to run asynchronously and simultaneously.
- Starting a new thread is very easy. When you create an instance of a std::thread, it will automatically be started.
- When you create a thread you have to give it the code it will execute. The first choice for that, is to pass it a function pointer.
- Calling join() function forces the current thread to wait for the other one (in this case, the main thread has to wait for the thread t1 to finish). If you omit this call, the result is undefined.

C++ Mutex

- The C+11 standard provides std::mutex primitive. A mutex object also provides member functions – lock() and unlock( ) - to explicitly lock or unlock a mutex. The most common use of a mutex is when one wants to protect a particular block of code. To this end the C++ standard library provides the std::lock_guard<> template

References

- people.ds.cam.ac.uk/nmm1/OpenMP/
- https://computing.llnl.gov/tutorials/openMP/