Livelock Solutions

Livelock

- What is meant by the term livelock?
  - A thread is livelocked when it is able to run, but cannot make any progress
- How does livelock differ from deadlock?
  - In deadlock, the thread cannot run at all
- Briefly describe a situation where livelock can occur
  - Livelock can occur when trying to avoid deadlock
  - Instead of blocking indefinitely when they cannot get a lock, the threads wait and retry
  - If the lock is not available, the threads will retry indefinitely
  - The threads are able to run, but cannot make any progress

Livelock Example

- Write a program which demonstrates livelock
- Suggest how livelock could be avoided in your program
  - Use scoped_lock to acquire both locks, or unique_lock with lock()
- Reimplement your program so it is not affected by livelock

Thread Priority

- What is meant by thread priority?
  - The priority of a thread is a number assigned to it by the operating system
  - A thread with high priority will be scheduled to run more often
  - A thread with low priority will sleep or be interrupted more often
- Does C++ support thread priority?
  - C++ does not directly support thread priority
  - Thread priority can be set by calling an operating system API
  - The native_handle member function of the std::thread object will return the data which is needed for this call

Resource Starvation

- What is meant by resource starvation?
  - Resource starvation occurs when a thread cannot get the resources it needs to run
- Give some examples
  - Deadlock and livelock, in which threads cannot obtain locks they need
  - Insufficient operating system resources which prevent threads from starting
  - In a badly designed scheduler, a low priority thread does not run often enough because higher priority threads monopolize the processor