```
#include <mutex>
// Modified by Craig Scratchley.  October 2021

/* // commented out by Craig Scratchley
class some_big_object
{};

void swap(some_big_object& lhs,some_big_object& rhs)
{}
*/

typedef int some_big_object; // added by Craig Scratchley

class X
{
private:
   some_big_object some_detail;
   mutable std::mutex m;
public:
   X(some_big_object const& sd):some_detail(sd){}

   friend void swap(X& lhs, X& rhs)
   {
      if(&lhs==&rhs)
         return;
      std::unique_lock lock_a(lhs.m,std::defer_lock);
      std::unique_lock lock_b(rhs.m,std::defer_lock);
      std::lock(lock_a,lock_b);
      std::swap(lhs.some_detail,rhs.some_detail);
   }
};

int main()
{
   // added by Craig Scratchley
   // create other threads to form multi-threaded program...

   X x1(3);
   X x2(4);
   swap(x1, x2);
}
```

more flexible compared to lock_guard

chap 3.2.6

// not lock right away, until required

lock required

// lock func

// avoid deadlock

chap 3.2.5.

you don't want 2 threads run halfway, you want at least 1 thread finish