~~Chap 5~~

Skip monitor & message passing

Readers / Writers Problem (custom synchronization)

- shared data among threads (file, queue etc.)
- threads that only read data
- threads that only write data

① Any # of readers may simultaneously read file
② only 1 writer at a time
③ if a writer is writing to the data, no reader may access it

Many readers with no writer
1 writer no readers

readers do not write
writers do not read

- mutex around all? expensive to aquire, especially if no writers -

- For instance, suppose shared data is a library catalog.
  patrons read catalog to find book
  librarian writes catalog to add / subtract books
  - if mutex around all, only 1 person can check catalog at a time, every other reader waits

- we want all readers to have concurrent access until 1 writer, then blocked

```
if (count ==0)
  cv.notify_one
```

1st

light switch problem.
1st person in room turns on switch
lots of other people come in
last person in the room turns it off

```
if firstperson in                 if firstreader in
    turn on lights        =>        (lock out writers)

if last person out                if last reader out
    turn off lights                   allow writers
```

```
int count=0;
mutex mwriter //locks out writers
mutex mcnt; //locks count access

void reader(){
    while (true){
        mcnt.lock();
        count++;
        if (count:7 1)
            mwriter.lock() //lockoutwriters
        mcnt.unlock();
        //lots of very lengthy calcs+code

        mcnt.lock();
        count--;
        if (count ==0)
            mwriter.unlock() //allow writers
        mcnt.unlock(); }
```

=> only locking when just readers not whole func

```
void writer()
    while (true) {
        mwriter.lock();
            ⋮
        write ops here
        mwriter.unlock
```

lock the ~~whole~~ thing

see 410_readers_writers_mutexes.git