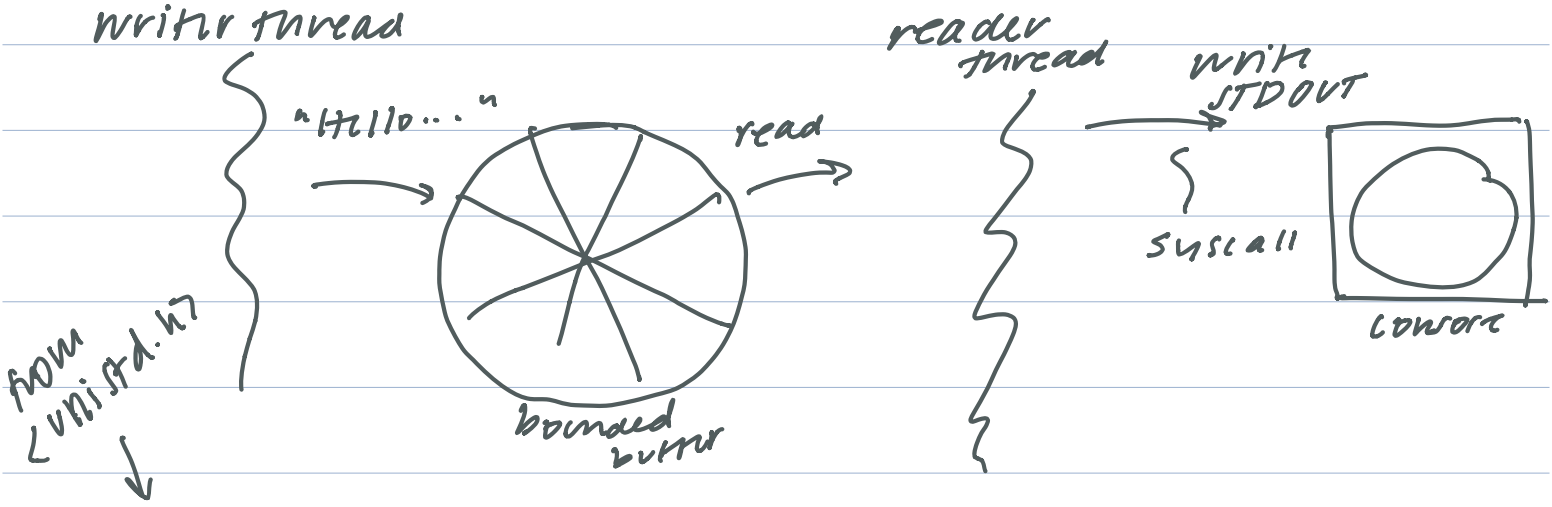
CS61 Lecture 26 — December 3

- redo due on the 13th

Polling



VSleep(1000000) makes writer poll

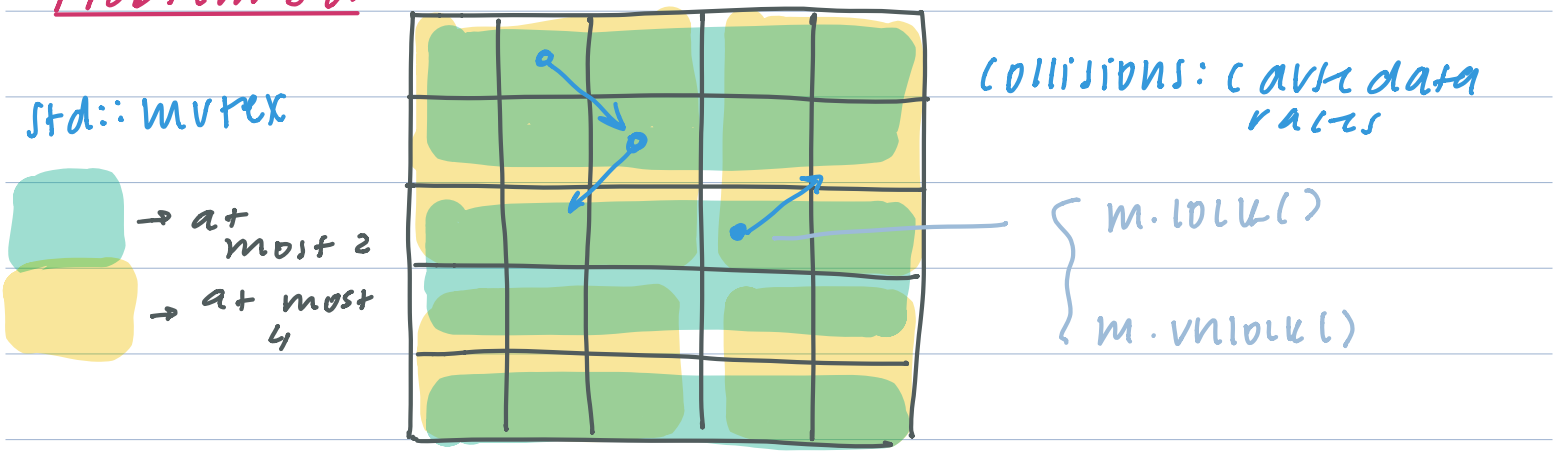
GDB

info threads: lists all threads and where they are blocked

thread 3: switches context to thread 3

bt: backtrace

Problem Set



- look locally around the ball
- cannot see where ball will end up → need to obtain all mutexs at the beginning
- only 2 balls need to move at the same time
- can have overlapping regions

Condition Variables

polling code versus blocking code

↳ correct but inefficient

Read

when read begins, it is 0, fun
1. block until either · there is at least 1 char in buf
· but is closed for writing

Fair

· ensures synchronization objects are evenly distributed amongst threads in system

speed ↔ fairness tradeoff

```
while ( ... ) {
    this -> mutex.unlock()
    sched_yield();   ← not enough for fair sharing
    this -> mutex.lock()
}
```

```
+ global
    std::condition_variable nonempty_;
+ in method
    std::unique_lock<std::mutex> lock(this->mutex_);

    while ( ... ) {
        this -> nonempty_.wait(lock);
    }
```

block until condition changes ← block until nonempty

```
+ every place the condition might change
    this -> nonempty_.notify_all();
```

std::condition_variable

① represent blocking for a "condition"

→ buffer not empty

→ key was pressed

② avoids sleep-wakeup race condition by integrating with mutex

wait(std::unique_lock<std::mutex> &l)

- blocks until notify
- l must be locked on entry
- l is unlocked while blocking

notify-all()

- unblocks all waiting threads