9/13/2010

Midterm #1
September 22 -
Wednesday
Room TBD

1. When bagging food (OrderTaker) - that wasn't ready @ order time - they bag any order that can be bagged until:

   - no more orders to bag
   - remaining orders don't have all food cooked

2. For an eat in Customer, does the OrderTaker wait for a waiter to be available before taking another order?

   No. Once the OrderTaker gives an order number to the Customer & makes it available to a waiter (or where can find it), they take the next Customer.

Managers wake up waiters.
Managers never wait on a CV, unless they are taking an order.

Situation: All waiters are on break.

{ An eat-in Customer gives an order.
Manager does not bag the order; an OrderTaker bagged it.

Question: How will the Manager know to wake up a waiter?

Manager must have code:

```
if ( /* a waiter is needed */)
    wake up a waiter
```

A test

Customers who want to eat-in & the restaurant is full, must wait.

Point #1: This test does not require Customers be seated.

Point #2: Must have eat-in Customers
Solution: make all Customers eat-in

Point #3: Restaurant must be full
Solution: Make the restaurant "full" @ ⊗ test initiating

Point #4: What entities do I need for this test?
- Customers - more than 1
- OrderTaker - 1 is fine

Methodology for writing multi-threaded interactions

"Write" one at a time - only 2 entities

1- Analysis
   - lock
   - 1 or more CVs
   - monitor variables

   } give them names

2. Design, ending up w/ pseudocode

Ignore any other entities

3. Write code for both entities

4. Compile the code

5. Test the code you just wrote

Customers wanting to place an order

- Customers must decide whether to ~~get~~ wait in line or go right up to an OrderTaker
  - Must be some monitor variable(s) for OrderTaker status - use an array (of ints)
    orderTakerStatus
    - I used: 0 for available
      1 for busy

Question: How does an OrderTaker know which status belongs to them?

Use Nachos

```
for(int i=0; i<5; i++){
    Thread *t = new Thread("—");
    t->Fork(OrderTaker, (i));
}
```

```
void OrderTaker(int myIndex){
```

- When customers have to wait in line - use a CV
  custWaitingCV

- Must have a lock
  custLineLock

How would an OrderTaker know if are any Customers in line?
- use a monitor variable for line length - custLineLength

Issue: What value do monitor variables start with?
- custLineLength = ∅
- orderTakerStatus = {1} busy

Customer Pseudocode

```
void Customer (int myNumber) {
    int myOrderTaker = -1;        // local to each Customer
                                  // I don't have an OT, yet

    custLineLock->Acquire();
    for (/*maxNumber of OTs*/) {
        if (orderTakerStatus[i] == 0) {   // available
            orderTakerStatus[i] = 1;      // busy
            myOrderTaker = i;
            break;
        }
    }
    if (myOrderTaker == -1) {
        // No available OT
        custLineLength++;
        custWaitingCV->Wait(custLineLock);
    } // An OT signaled me - find them

    // I have an OrderTaker
    custLineLock->Release();
```

CSCI402 Lecture September 9, 2010

Analysis for Customer-OrderTaker Interaction)

- 1 line for Customers waiting to give order
- Multiple Customers
- 1 or more OrderTakers
- Cust wait in line if no available OrderTakers

Data

- Need monitor variable to track state of OrderTakers

```
Condition* custWaitingCV int custLineLength //monitor
variable OT can check to see if he has to signal next
thread)
Lock* custLineLock //prompt user for maxNumOrderTakers
(don't want to make this constant)
int orderTakerStatus[maxNumOrderTakers] //1st monitor
variable customer checks when he enters the restaurant
- initialize to "BUSY" (1=BUSY, 0=FREE) (or use an enum**)
int myOrderTaker = -1; //need to index your orderTaker
- initialize all variablefirst, THEN make threads
```

Customer Pseudocode

```
custLineLock->Acquire();
for(maxNumOrderTakers) {
  if(orderTakerStatus[i]==0) {
    OrderTakerStatus[i]=1;
    myOrderTaker=i;
   }
}
if(myOrderTaker == -1) { //no one's available, get in line
   custLineLength++;
   custWaitingCV->Wait(custLineLock); //wait for OT to be
available
}
/*potential problem: if OT changes status to available, I
go to end of ready queue, while a customer who just started
might search, find available OT, and take him without
waiting in line→ not fair! We need another status*/
//get a "waiting" OT w/ different value, i.e. 2
for(maxNumOrderTakers) {
   if(orderTakerStatus[i]==2) { //note: if this condition
    fails, we know it must be a manager doing the order
    taking
    myOrderTaker = i;
   }
   orderTakerStatus[i] = 1;
}
//custLineLength--; //no longer decrementing here
//@ this point, customer is DONE waiting in line and is
```

Customer pseudocode

```
ready to give order
custLineLock->Release(); //time to play some tennis
OrderTakerLock[myOrderTaker]->Acquire(); //give ordertaker
my order
OrderTakerCV[myOrderTaker]>Signal(orderTakerLock[myOrderTak
er]); //monitor variable for each OrderTaker, index
myOrderTaker
orderTakerCV[myOrderTaker]->Wait();
```

OrderTaker Pseudocode

```
void OrderTaker(int myId); //how we're keeping track of
diff OTs
while(true) {
    custLineLock->Acquire();
    if(custLineLength > 0) {
     custWaitingCV->Signal(custLineLock);
     custLineLength--; //now let the OrderTakers decrement
the line length
     orderTakerStatus[myId] = 2;
    else if(/*food to bag*/) {
     custLineLock->Release(); //bag 1 order at a time
continue;
    } else {
     //Nothing to do
      orderTakerStatus[myIndex]=0;
    }
}
//Now I need to wait for a customer
orderTakerLock[myId]->Acquire(); //array of locks -
guarantees I have control of the monitor variable before
the customer does
custLineLock->Release();
orderTakerCV[myId]->Wait(orderTakerLock[myId]);
//Now process the order..}
```

cust in line

or bag food

or

bag food

Can I can use one CV orderTakerCV [myId] for the entire order taking process