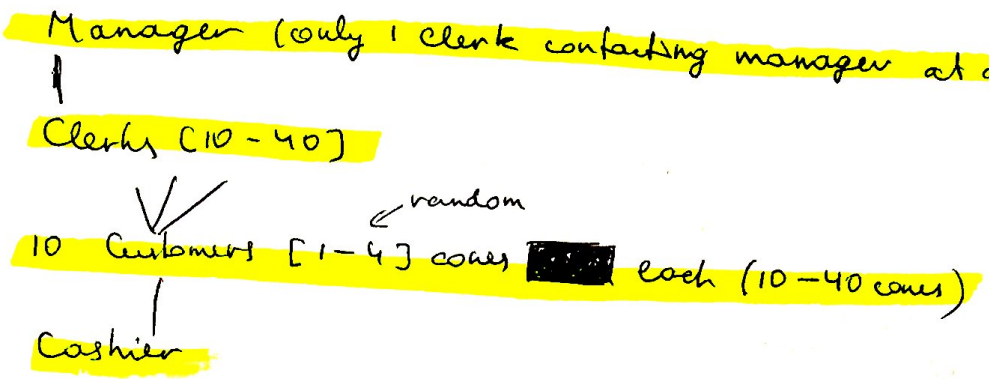
CS107 Lecture 18



challenge: maintain synchronization between threads, ~52 thread

Scenarios:

1) a customer asks for n coins, waits until all n clerks give each 1 coin, then goes to the cashier
   ↳ 1 to N rendezvous pattern
2) only one clerk contacting manager at any time
   ↳ binary lock pattern
3) Manager waits until all potentially 40 coins were served, before leaving
   1 to N rendezvous pattern
4) Manager waits for clerk to request,
   clerk waits for manager to approve/disapprove
   1 to 1 rendezvous pattern
5) Customers are served by Cashier in FIFO order.
   Cashier waits until Customer shows up in line, Customer waits for Cashier to finish handling payment
   1 to 1 rendezvous pattern

```
int main(___, ___)   // spawning all top-level threads
{
  int totalCones = 0;
  InitThreadPackage();
  SetupSemaphores();  // initialize global semaphores
  for (int i=0; i<10; i++)
  {
    int numCones = RandomInteger(1, 4);
    ThreadNew(___, Customer, 1, numCones);   // ___ = debug name
    totalCones += numCones;
  }
  ThreadNew(___, Cashier, 0);
  ThreadNew(___, Manager, 1, totalCones);
  RunAllThreads();
  FreeSemaphores();
  return 0;
}
```

not waiting here

manager is waiting until all cones are served and paid for

// global Semaphore variables, packaged in structs

```
struct inspection {
  bool passed;          (false)
  Semaphore requested;  (0)
  Semaphore finished;   (0)
  Semaphore lock;       (1)
}
```

initialize in SetupSemaphores

1-to-1 rendezvous

manager waits until inspection requested

clerk waits until inspection finished

clerk / manger lock, [redacted]

```
struct line {
  int number;           (0)
  Semaphore requested;  (0)
  Semaphore customers[10];
  Semaphore lock;       (1)
}
```

next place available in FIFO line

1-to-1 rendezvous

cashier waits until requested

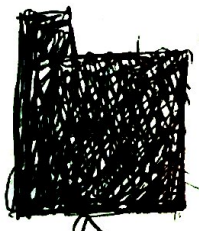
each customer waits until payment processed

need array due to FIFO constraint

customer / cashier lock [redacted] (customer #)

```
void Manager (int totalCones Needed)
{
    int numApproved = 0;
    int numInspected = 0;
    while (numApproved < totalCones Needed)
    {
        Semaphore Wait (inspection.requested);
        numInspected ++;
        inspection.passed = RandomChance (0, 1);
        if (inspection.passed) {
            numApproved ++;
        }
        Semaphore Signal (inspection.finished);
    }
}

void Clerk (Semaphore semaToSignal)
{
    bool passed = false;
    while (!passed) {        // keep making cones, until a cone passes inspection
        makeCone();
        Semaphore Wait (inspection.lock);
        Semaphore Signal (inspection.requested);
        Semaphore Wait (inspection.finished);
        passed = inspection.passed;
        Semaphore Signal (inspection.lock);
    }
    Semaphore Signal (semaToSignal);
}
```



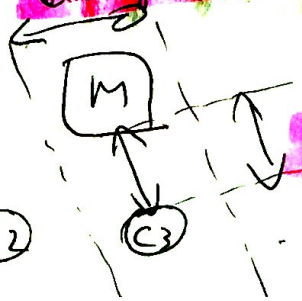
only one clerk makes request at a time

critical region

1 to 1 communication between clerk and manager threads

1 to N comm. between clerks and customer

binary lock



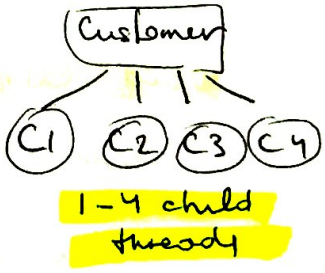
my comment

```
void Customer (int numCons)
{
    Browse();                                   // local for a customer instance
    Semaphore clerksDone; (0)
    for (int i=0 ; i < numCons ; i++)
    {
        ThreadNew (—, Clerk, 1, clerksDone);
    }
    for (int i=0; i < numCons ; i++)
    {
        SemaphoreWait (clerksDone);
    }
    SemaphoreFree (clerksDone);
    WalkTo Cashier ();
    SemaphoreWait (line.lock);
    int place = line.number++;
    SemaphoreSignal (lock);
    SemaphoreSignal (line.requested);
    SemaphoreWait (line.customers[place]); }
void Cashier()
{
    for (int i=0; i<10; i++)
    {
        SemaphoreW(line.requested); // when requested, there is the next customer with next number
        checkout (i);
        SemaphoreSignal (line.Customers[i]);
    }
}
```

1 to N pattern to wait until child threads are complete with respect to this customer thread

ready to pay

lock makes sure no other customer has the same #



1-4 child threads

my comment

coordination pattern

- Binary lock: selectivity of communication; which thread with which thread
- 1-to-1 rendezvous: correctness of communication

e.g. 1) clerk/manager
2) customer/cashier