CS 181AI
Lecture 10

Synchronization cont.

Arthi Padmanabhan

Feb 20, 2023

Logistics

- Assignment 3 due Friday
    - We’ll look at it for a bit today. Start early!
- Will start posting board notes

Last Time

- Locking & Synchronization
    - Mutex lock
    - Semaphore
    - Barrier
    - Started conditional variables

Today

- Finish locking: conditional variables, deadlocks
- Start looking at Assignment 3

Monitors & Condition Variables

- Wait & Notify
  - Wait: block myself and give up control of the lock (a queue is formed on this variable)
  - Notify: causes next thread in that queue to be released so it can re-acquire the lock and keep running

Example

```
class Keep_count
        count = 0
        lock = Lock()
        above_zero <- conditional variable
```

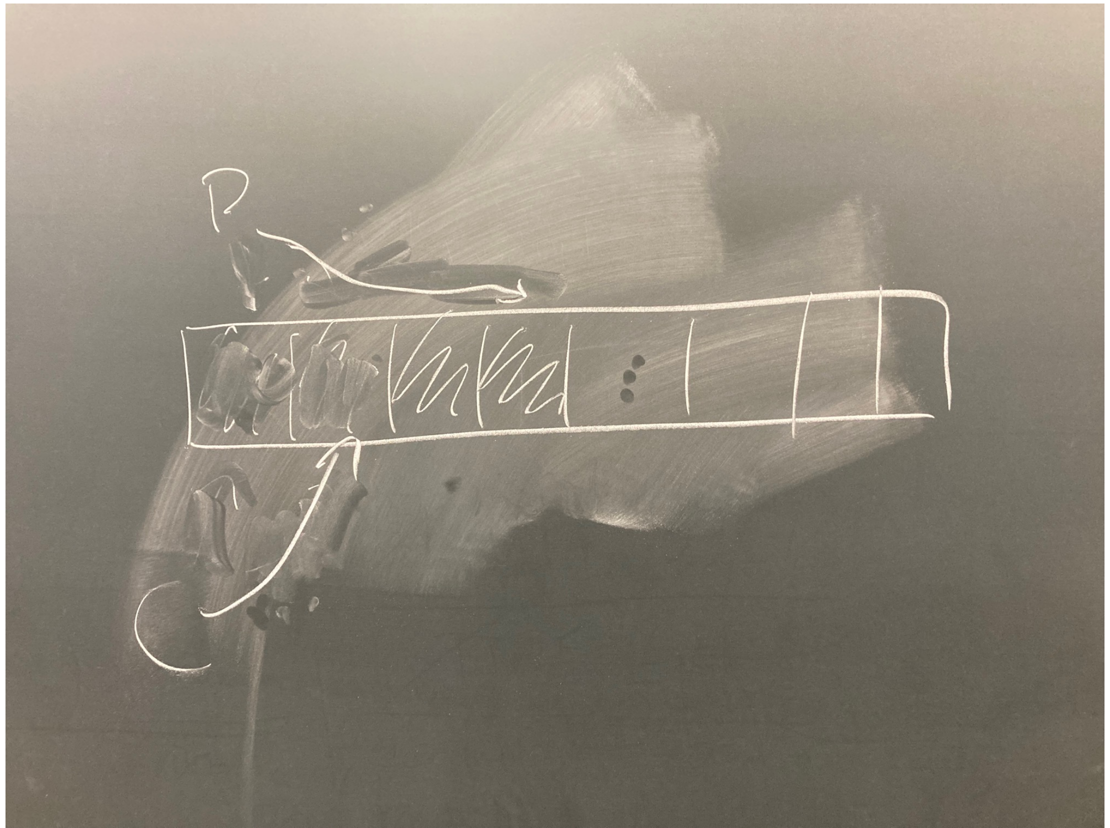
```
def print_when_greater:
        acquire lock
        while count <= 0:
                wait(above_zero, lock)
        print(count)
        release lock
```

```
def increment:
        acquire lock
        count += 1
        if count > 0:
                notify(above_zero)
        release lock
```

```
def decrement:
        acquire lock
        count -= 1
        release lock
```

Producer-Consumer

- Producer fills slots in a buffer
- Consumer consumed filled slot in the buffer



Example

Monitor Producer-Consumer:

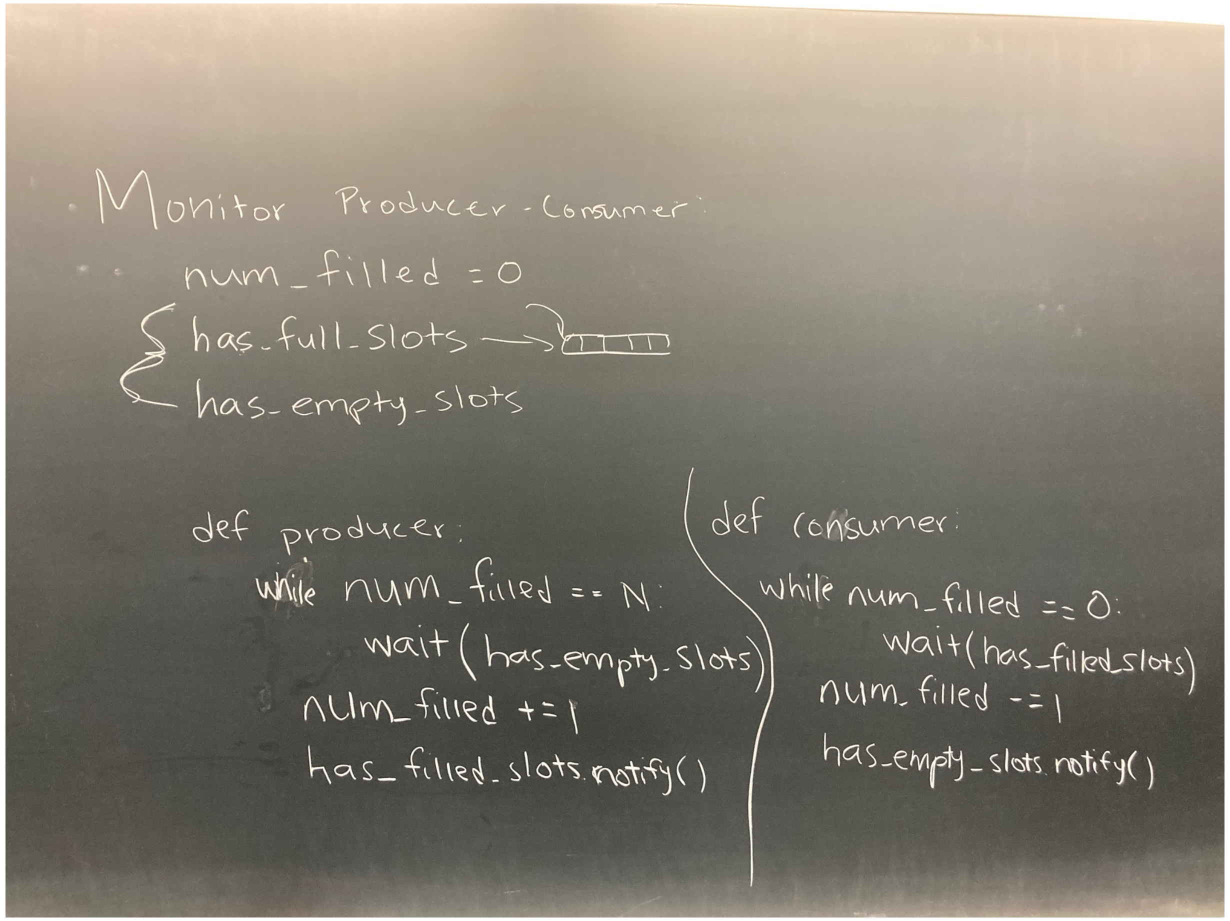
```
    num_filled = 0
    has_filled_slot
    has_empty_slots

    def producer():



    def consumer():
```

Board Work: Producer-Consumer



```
Monitor Producer-Consumer:
    num_filled = 0
    has_full_slots
    has_empty_slots

def producer:                           def consumer:
    while num_filled == N:                  while num_filled == 0:
        wait(has_empty_slots)                   wait(has_filled_slots)
    num_filled += 1                         num_filled -= 1
    has_filled_slots.notify()               has_empty_slots.notify()
```

Example

Monitor Producer-Consumer:

```
    num_filled = 0
    has_filled_slot
    has_empty_slots
    lock = Lock()

    def producer():
        while num_filled == N:
            wait(has_empty_slots,
lock)
        num_filled += 1
        has_filled_slots.notify()
```

```
def consumer():
    while num_filled == 0:
        wait(has_filled_slots, lo
    num_filled -= 1
    has_empty_slots.notify()
```

Bank Account Example Cont.

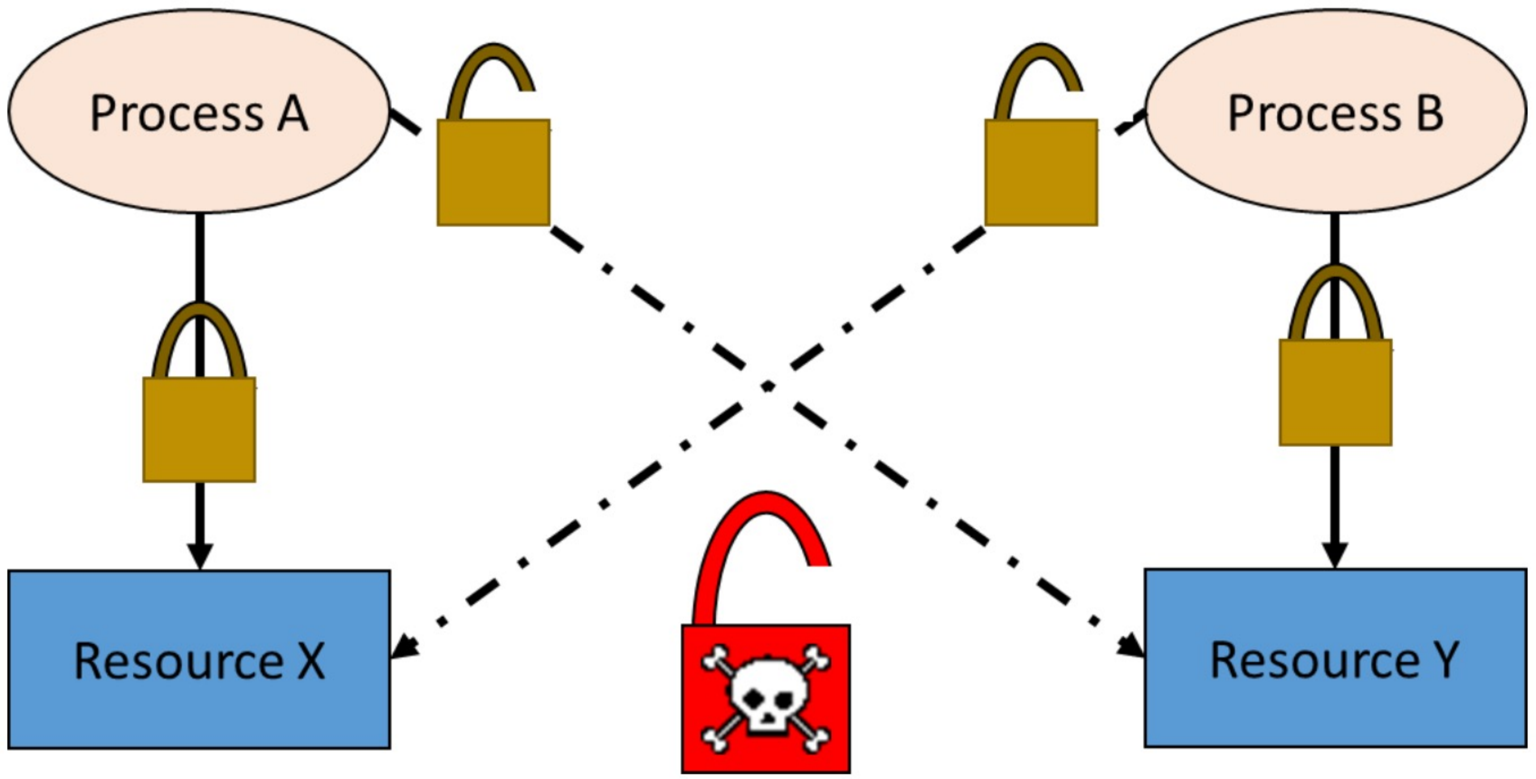
- User A wants to transfer money from Account 1 to Account 2

Bank Account Example Cont.

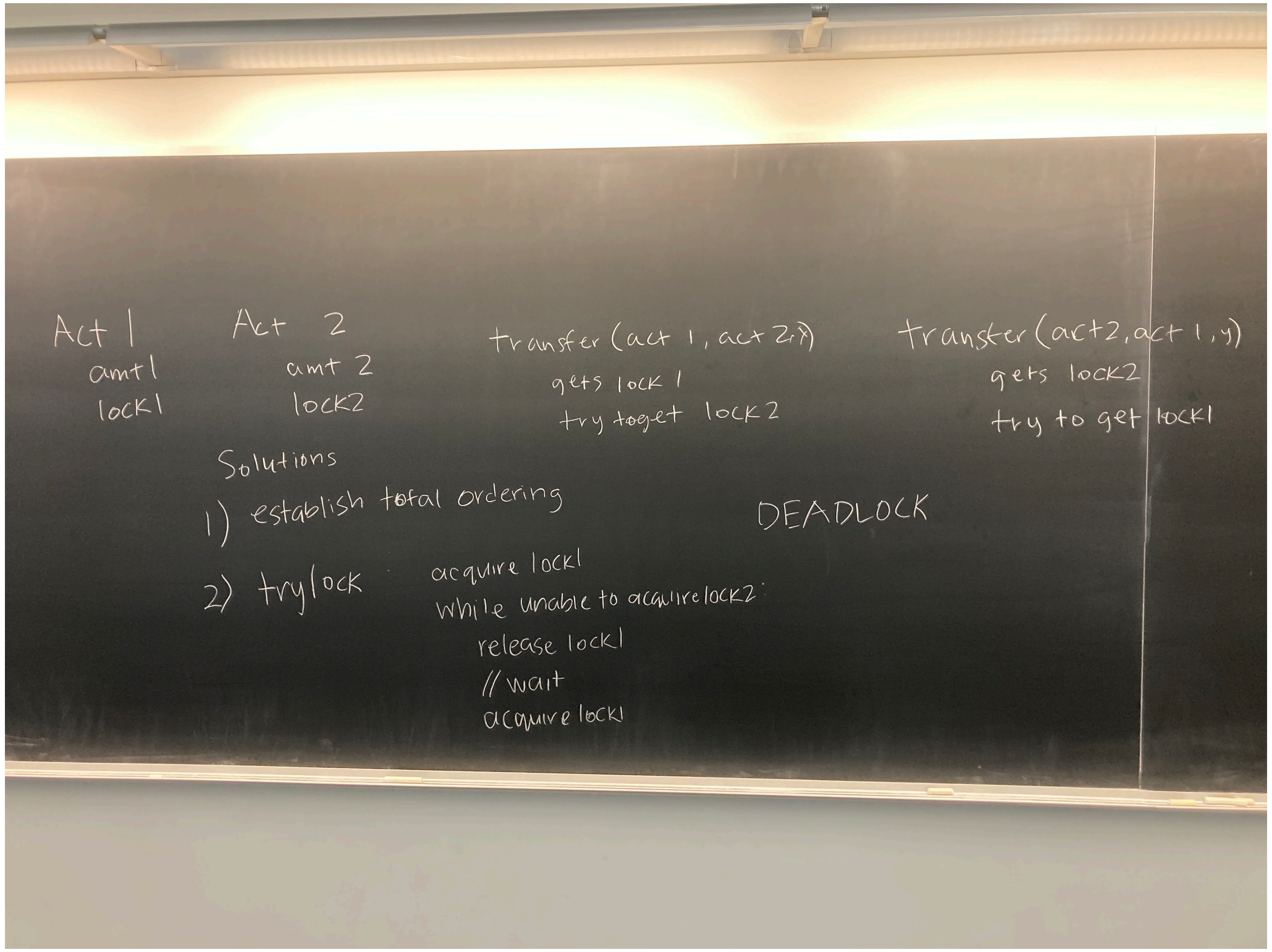
- User A wants to transfer money from Account 1 to Account 2
- User B wants to transfer money from Account 2 to Account 1

Deadlocks

- A deadlock occurs when none of the processes can make progress because there is a cycle in the resource requests



Board work: deadlocks + solutions



Act 1
amt1
lock1

Act 2
amt 2
lock2

transfer(act 1, act 2,x)
gets lock 1
try toget lock 2

transfer(act2, act 1, y)
gets lock2
try to get lock1

DEADLOCK

Solutions

1) establish total ordering

2) trylock

```
acquire lock1
while unable to acquire lock2:
    release lock1
    // wait
    acquire lock1
```

Deadlock Prevention

- Total ordering
- Trylock

Assignment 3

- Start looking at assignment and ask questions
- Goal: you have an idea of how you would approach the problems