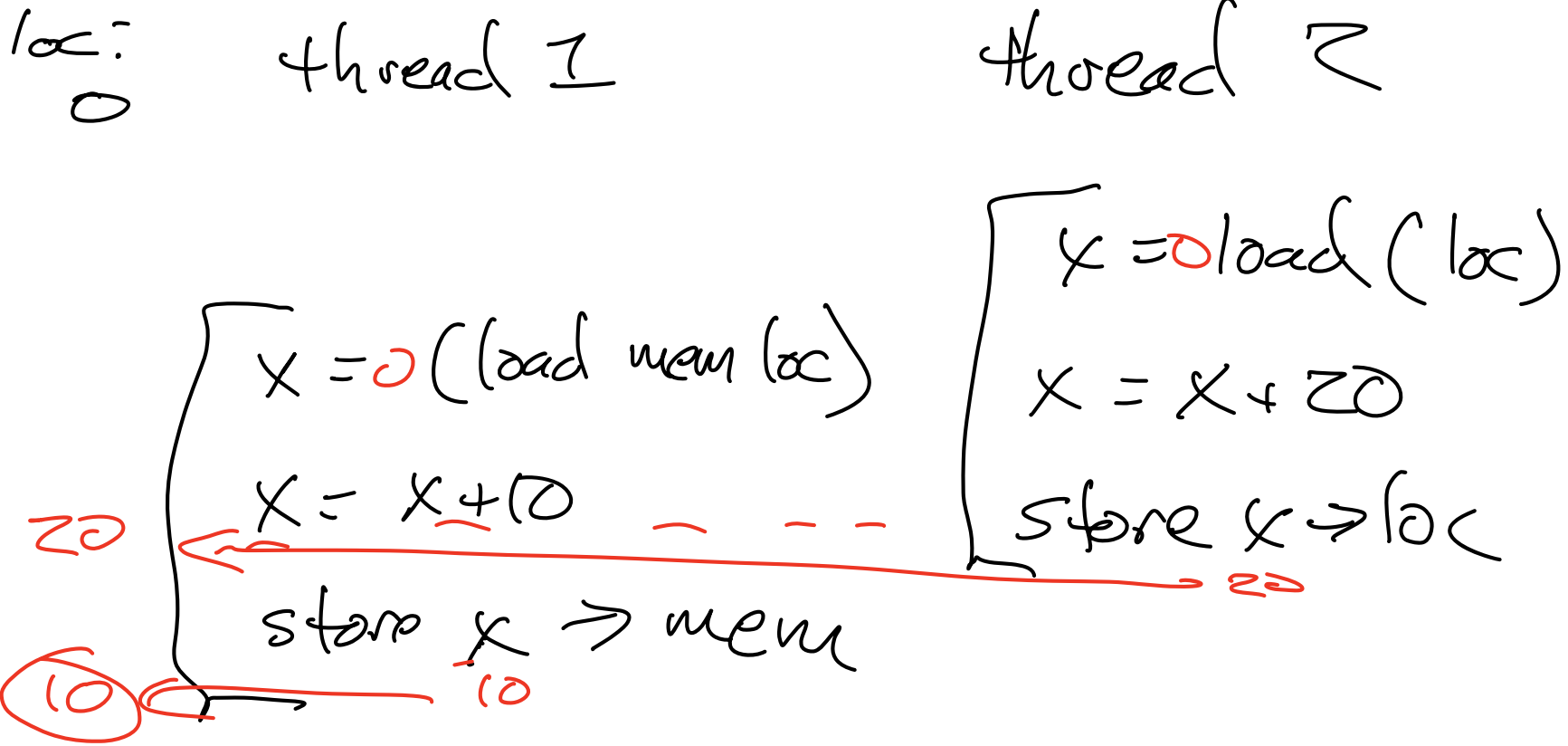


CS 3650 – Computer Systems
Spring 2024
Peter Desnoyers

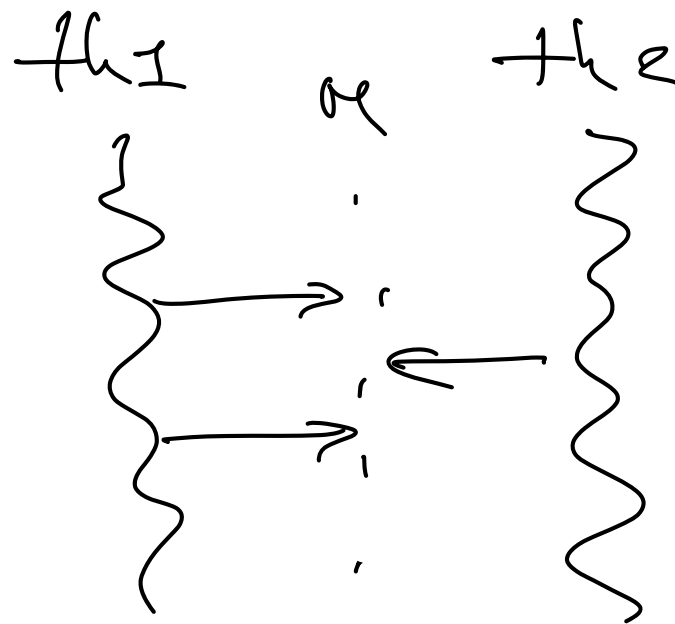
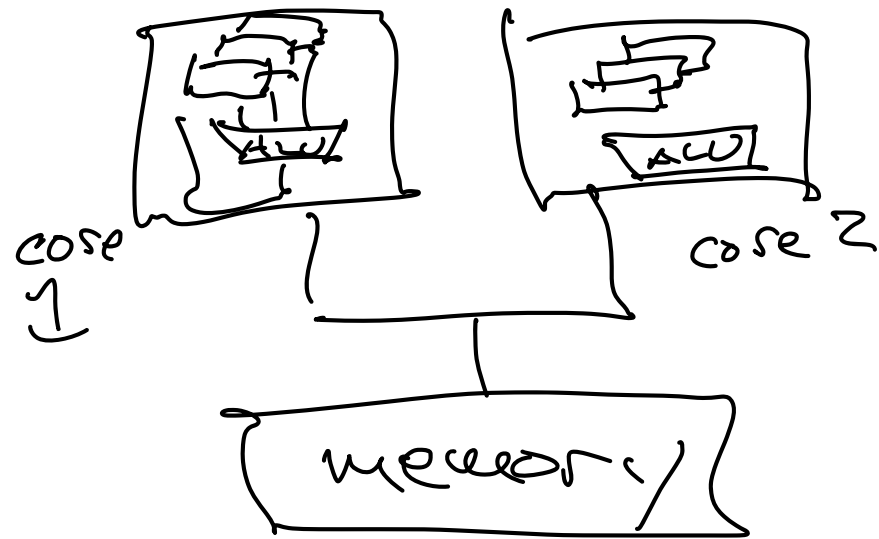
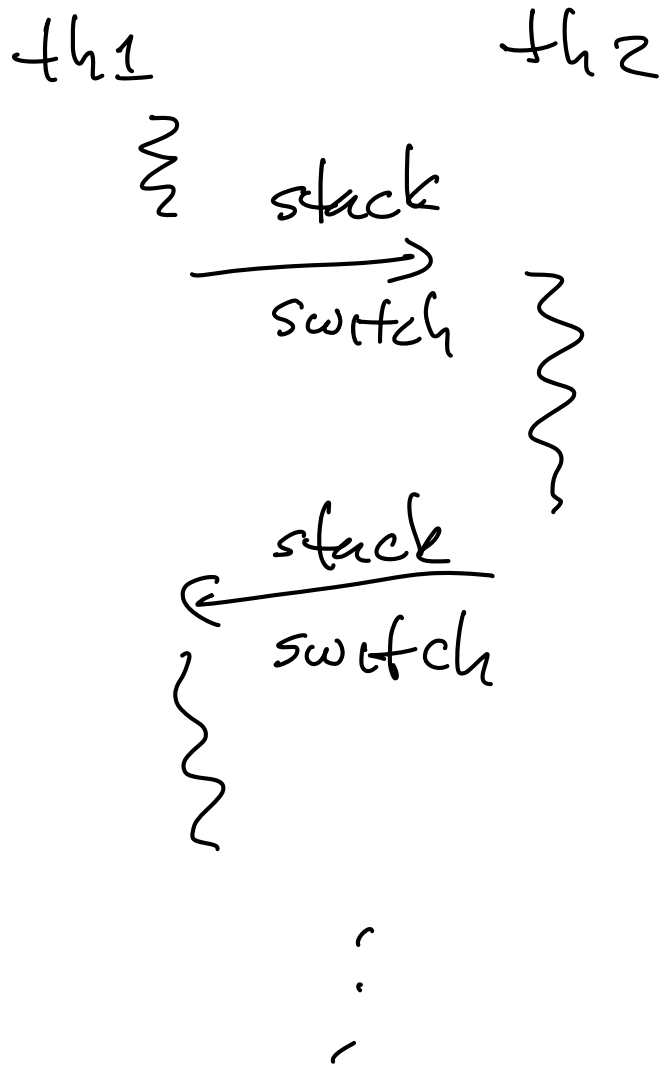
Lecture 13, Tue Feb 20 2024

Race conditions



~~30~~ 10

single core threads vs. multi-core



Why is multi-threaded hard?

→ unprotected data can change unexpectedly

thr 1

x = read(loc)
x += y
store x → loc

x = read(loc)
x += y
store x → loc

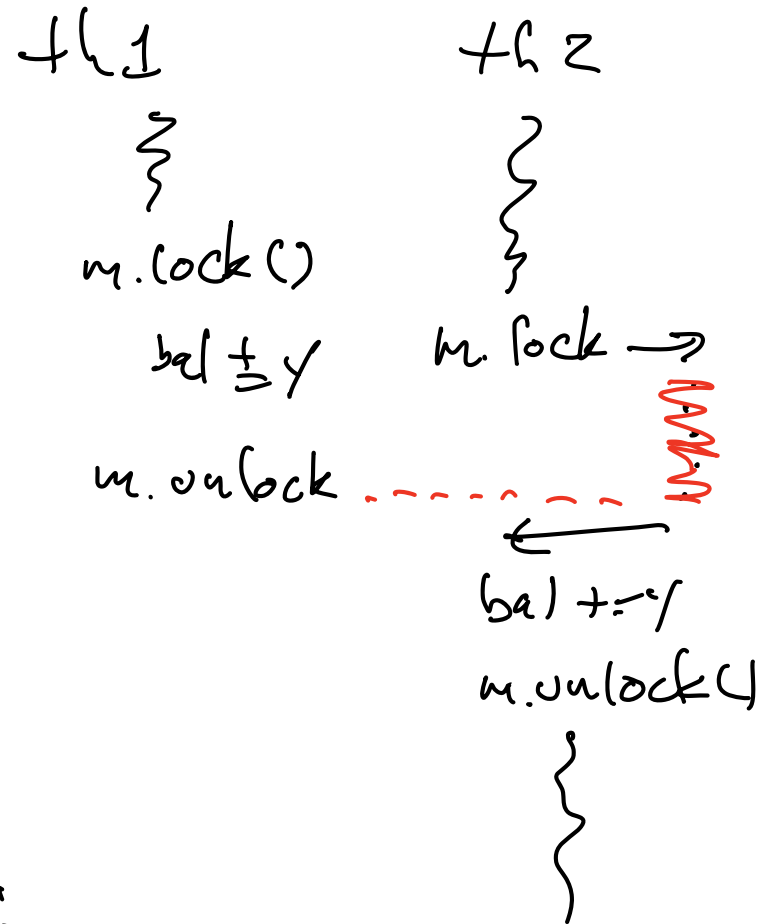
solution: mutex (mutual exclusion lock)
mutexes protect data

Bank account

```
class
  balance : int
  m : mutex
  method deposit (sum):
    m.lock()
    balance += sum
    m.unlock()
```

simple mutex mental model:
bool is_locked

method lock: while (locked) wait
locked = true



unlock:
locked = F

Single CPU

→ disable interrupts

→ + a brace (don't call switch on purpose, either)

mutex-lock

disable int

while locked
wait

locked = T

enable int

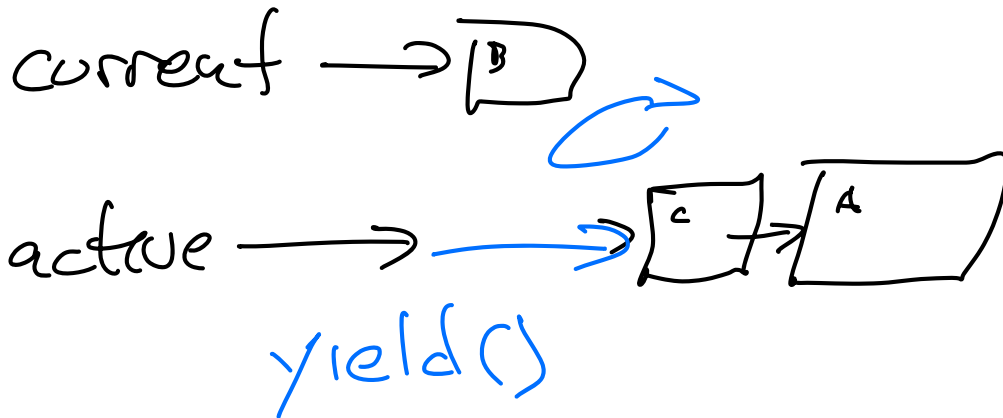
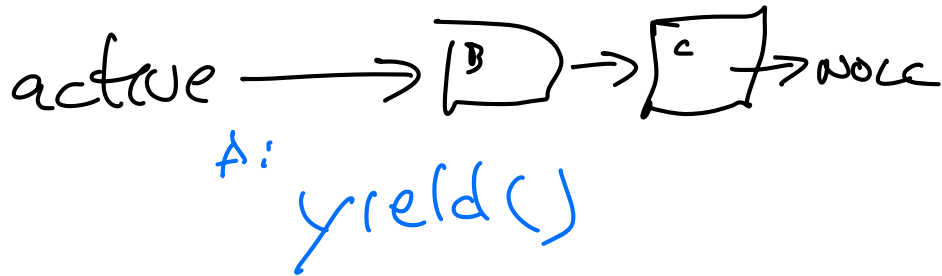
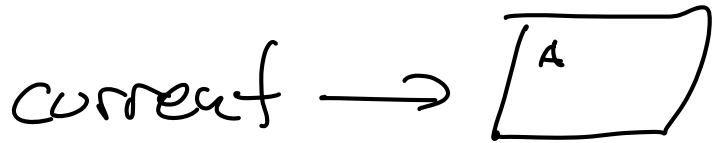
unlock:

disable

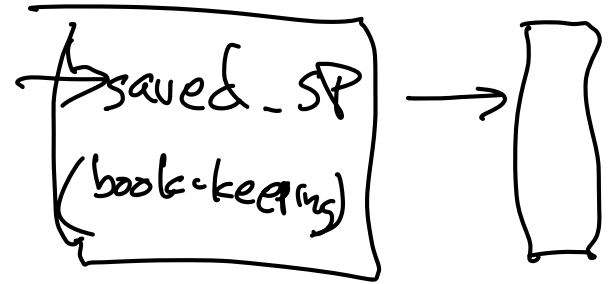
locked = F

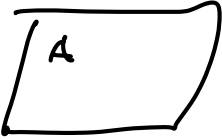
enable

Thread scheduling



thread struct:





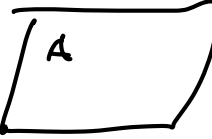
current →  ← thread control block

active →  →  → NULL

A: lock (locked mutex)



current → 

active →  → 

B: unlock()

~~some where else!~~

mutex:

{ bool is_locked
queue (thread CB): waiting

lock:

disable

if locked == T:

- remove from current
- top of active \Rightarrow current

append self \Rightarrow u. q

enable

switch \rightarrow current

else

locked = T

enable

current \rightarrow B

active \rightarrow C \rightarrow \emptyset

mutex:

locked: T

q: \rightarrow A \rightarrow \emptyset

\curvearrowright

C \rightarrow A

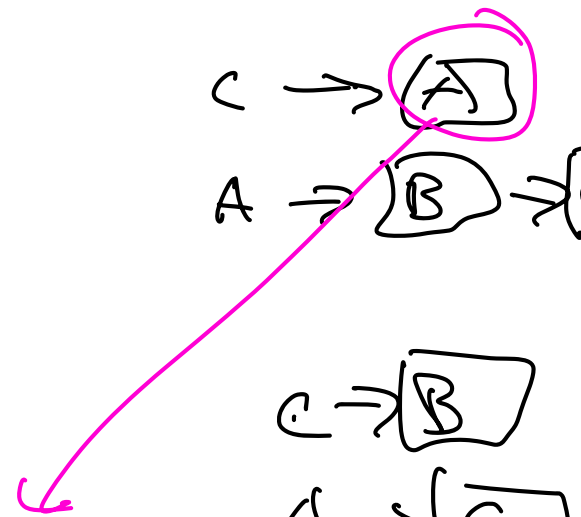
A \rightarrow B \rightarrow C

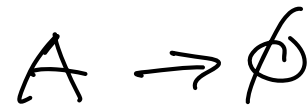
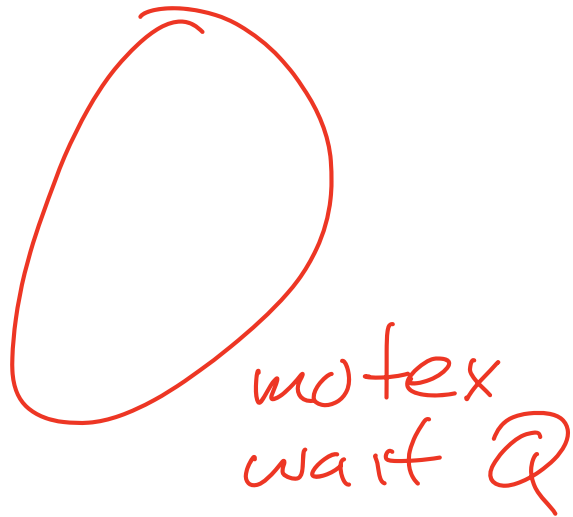
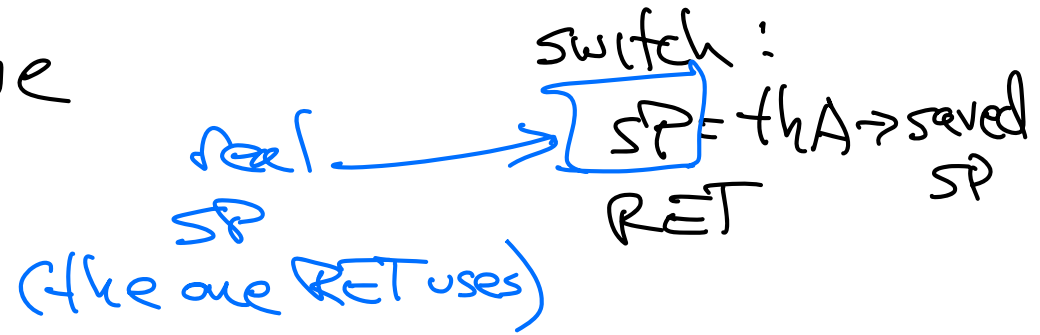
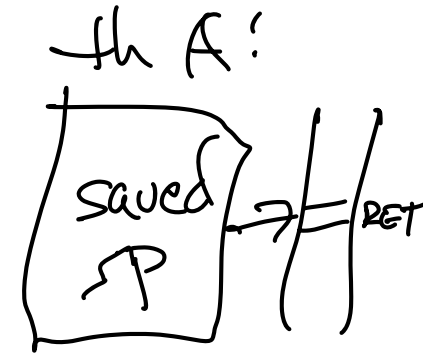
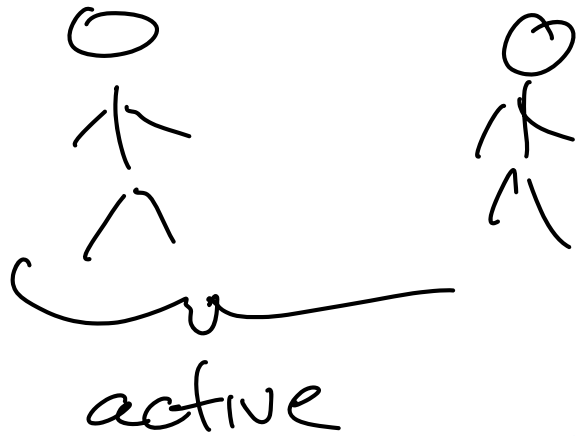
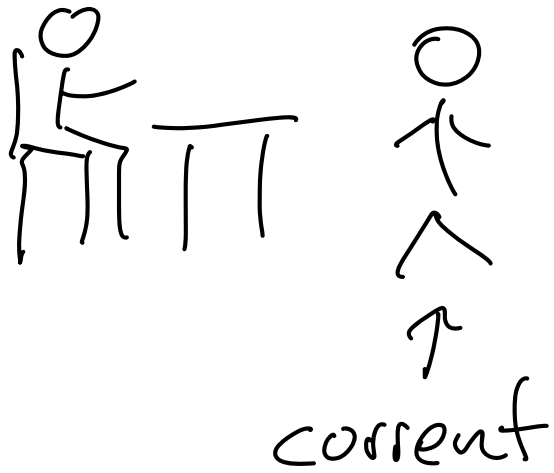
C \rightarrow B

A \rightarrow C

u:

q \rightarrow A





A: lock locked mutex

lock:

disable

if locked == T:

- remove from current
- top of active \Rightarrow current

append self \Rightarrow a. q

enable

switch \rightarrow current

else

locked = T

enable

unlock:

enable \rightarrow

if q not empty:

t = q.pop()

append t to active

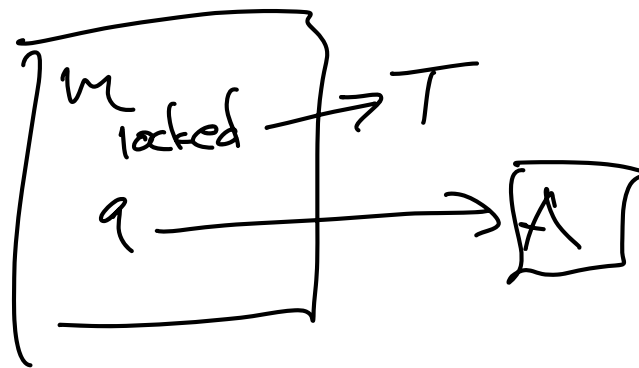
else

locked = F

enable \rightarrow

current \rightarrow

active \rightarrow



holte mutex:

lock if locked:

get out of line

let the next person go

else

locked = T

unlock: if someone is waiting in mutex Q:

put them back in line

else

locked = F

what do you do with a mutex?

protect data

class account
int balance
mutex m

A1 = new account

A2 = new account

th1

A1 → deposit()

x1004 balance

th2

A2 → deposit()

x1008 balance

class

int balance
m

method deposit

lock m : balance += sum

unlock

method withdraw

lock m -

unlock : balance -= sum

Deadlock

th1 lock(A)
 lock(B)
 ⋮
 unlock(B)
 unlock(A)

th2 lock(B)
 lock(A)
 ⋮
 unlock(A)
 unlock(B)

th1 : lock(A)
 lock(B) →
 unlock(A)

th2 : lock(B)
 lock(A)
 ⋮
 unlock(B)