[redacted]

chap 5 ①

[redacted] last part of chap 5 see by next tu

Multiprogramming - multiple processes single CPU

Process1 — —

Process2 — —

Process3 — —

Multiprocessing - multiple processes, multiple cores

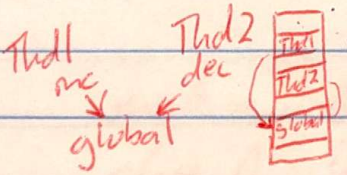
core1 p_1 — —
p_2 —

core2 p_3 — —
p_4 —

we note P1 & P3 concurrent

Distributed: multiple processors w/ multiple processes { difficult to coordinate, not discussed in this course }

① race condition: outcome depends on who gets done first, often tough to reproduce & debug



thread Atomic critsec.ppt

show thread prog with ~~critical sections~~ not included

② critical section - code that accesses shared resource that must complete w/o interrupt

thread ex. increment is the critical section but can be arbitrarily large.

Question: what if 100 threads ~~access~~ read a global

No you need to ~~protect it~~. worry about completing?

No, if never changed then no need ~~to protect~~

- atomic operation - sequence of 1 or more instructions. Must appear indivisible, no other process can see an intermediate state or interrupt it.

thread-atomic-criticalsection.cpp

show std::atomic<int> solution to above, show how mi int is 3 assembly; not interrupt any values... things go wrong. Demo.

```
getit   mov  eax dword ptr [global (0x8...)]
meit    add  eax, 1
storeit mov  dword ptr [global (0x8...)], eax.
```

int global = 0

lockadd1

Thd1 ... Thd2 ...

getit eax = 0 ⟶ getit eax = 0

meit eax = 0+1 = 1

storeit global = 1

meit eax = 0+1

storeit global = 1

what happened to 1+1 = 2?

show how atomics solve this: ~~But only~~

But works for single instructions only! what if you need to have a whole bunch of code ~~normal~~ atomic? takes much longer!

- single verses multi verses distributed processes
- race condition + atomic operations (w/ examples)
  atomics are single line only!
  critical section - code accesses shared resource that must complete w/out interrupt

  (will see more)

mutual

④ mutual exclusion - traffic cop as long as ~~you own the~~ everyone access using this. then only 1 can use at a time

global = 3

dec
int i ← getmutex
i = global
i = i + 1
global = i ← give up mutex

where is critical section

wait
block os
notifies when available

enter critical section

Thd1
Thd2
on 2 core system

exit crit section (Thd1 or Thd2 or both)

notice how things slow down in critical section (1 thread at a time)

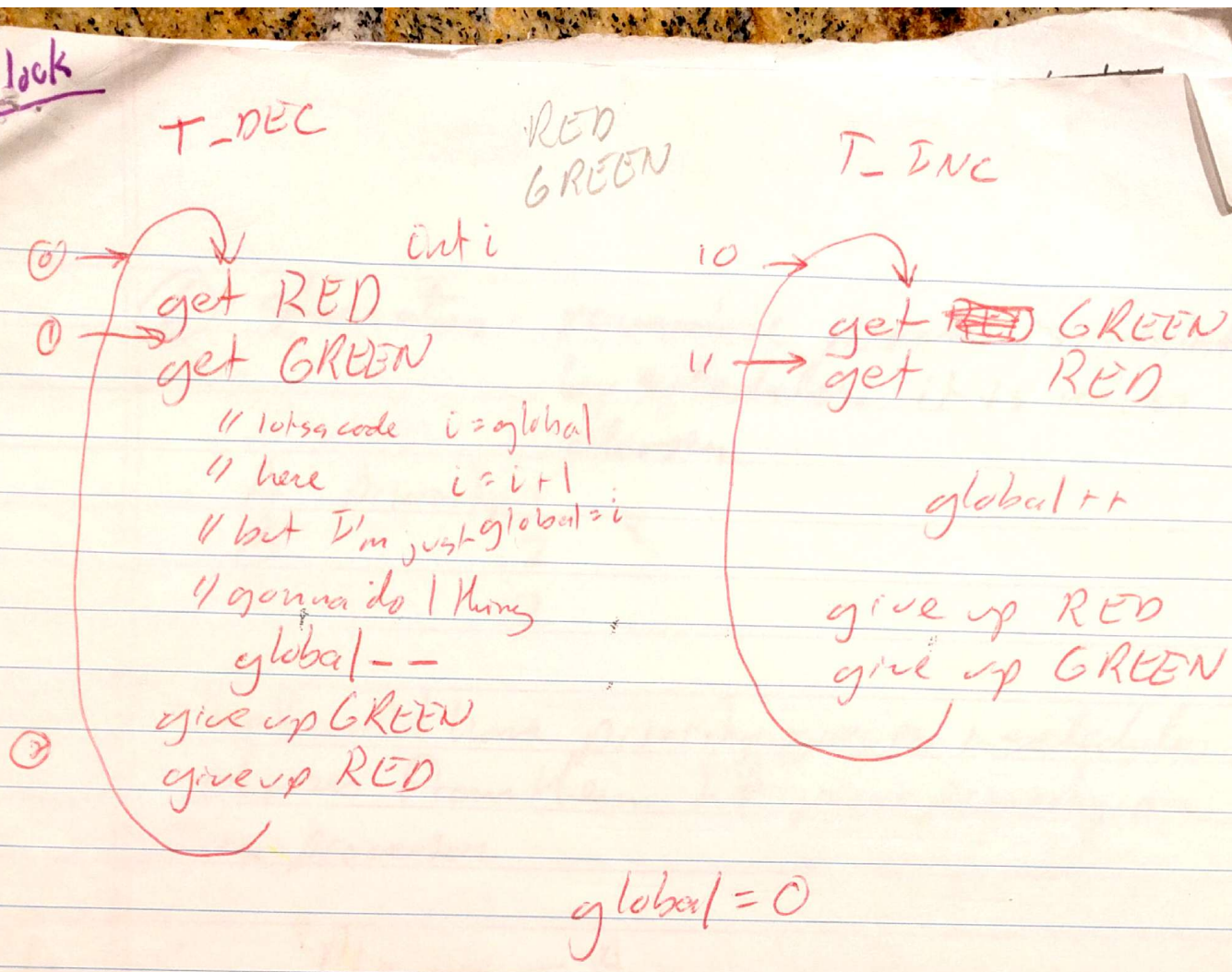
demo

so, it will slow things down.

⑤ Deadlock - 2 or more processes are waiting to proceed because each has something others wants.

do demo

deadlock



global = 0

- have it work a few times

- stop T_DEC @ (1) start T_inc @ (10)
  ~~stop~~ can't get red.

⑥ · Starvation - runnable process is overlooked by scheduler. it is never chosen.

A priority 1
B 2
C 2

real world have priority queues + scheduler chooses from them. IF pure priority on uniprocessor.

P1 queue – A
P2 queue – B – C

scheduler always chooses these. A will not run until B & C done.

(back to)

race condition - when program output depends on what gets done first, tough to reproduce & debug

ex.

```
int global = 2;

② void func() {
    if (global == 0)
        doZero(); ③
    else
        doFail(); ④
  }
  int main() {
    thread myt(func);
    global = 0;
  }
```

① critical section

race condition,
if ① happens before 2
then ③
else ④

who's gonna win?

(PSA - you may see code that "solves" this with delays.
(sleep-for(); sleep-until())
This is a non-scalable solution.)

Race Condition (Fnd:1)

```
int balance = 50;
1   void withdraw (int amount) {
2       if (balance > amount) {
3           cout << "approved";
4           balance -= amount;
5       }
    }
```

2 threads start w balance $50

T1 withdraw 40 } 65 > 50 !
T2 withdraw 25 }

Balance after	
50	T1 starts & gets to 3 (past the if)
50	preempted by T2,
25	T2 runs to end
-15	T1 switched back in finishes

go to slide 16

~~Hardware~~

Hardware - interrupt disable (slide 17)

10/31/17

Examined races, starvation, critical sections, deadlock
demoed mutexes (deadlock, non-reentrant)
showed how Hey effed code speed and correctness
~~show~~ how in the wrong places causes deadlock
also efficiency
Show them lock-guard next.