Today (3/6)

Concurrency

→ Locks ⇐ how to build?

(Data races)

→ Condition variable

(Control races)

Discussion: [P3] < 2 weeks

a, b

Data races:

examples:

shared counter

e.g. x = x + 1;

list insert

Solved: Locks

locks: a little control over what runs

Lock: construction

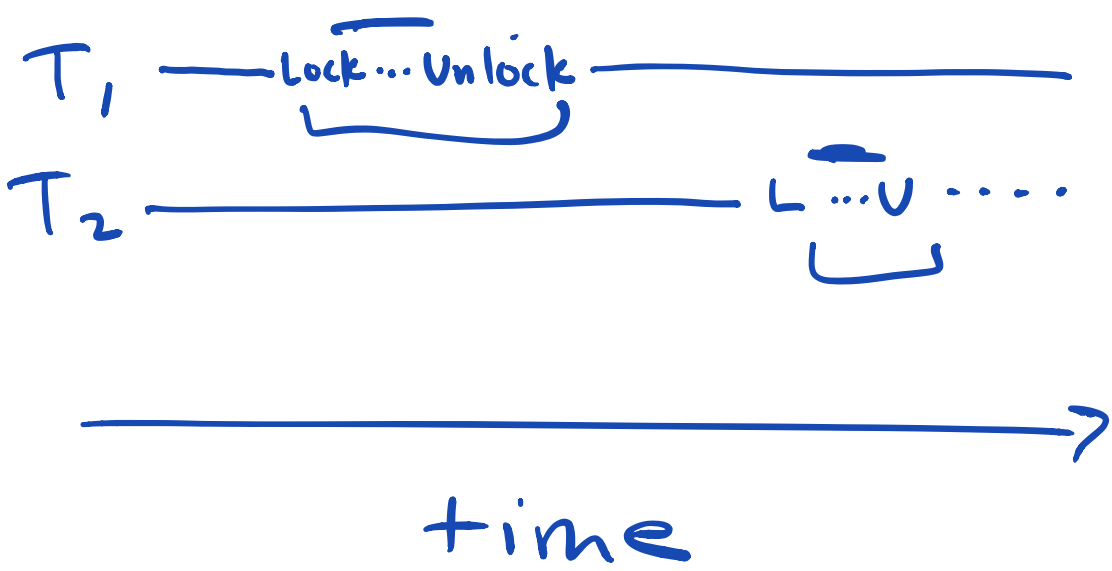
=> Simple spin lock

correct, "it works"

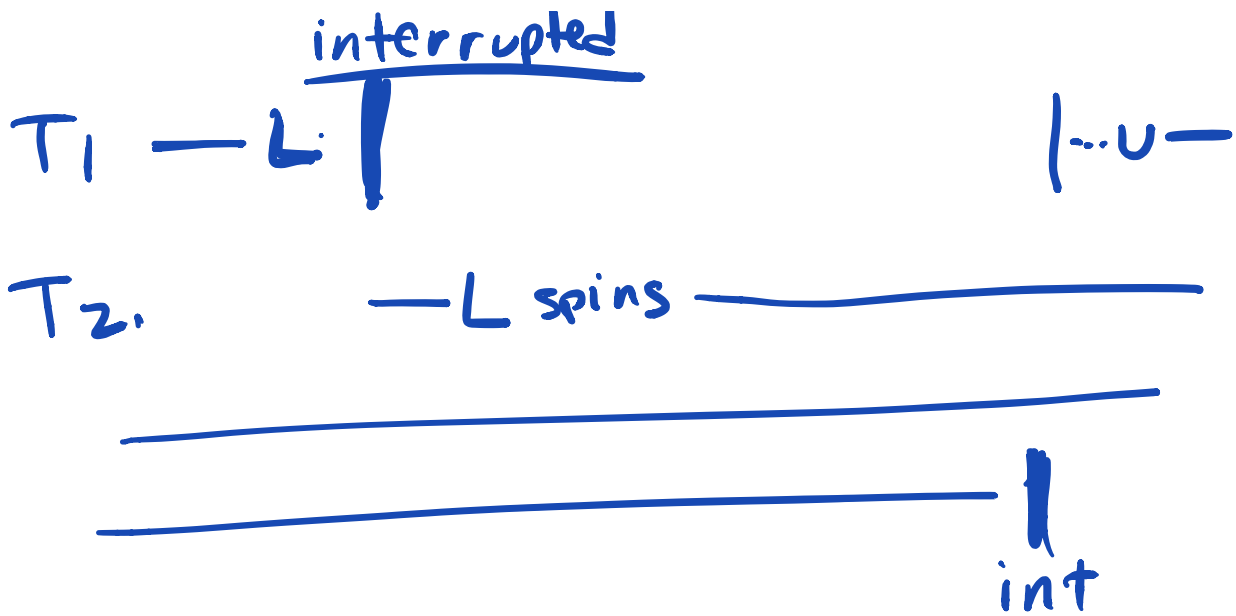
Problems:

→ [Too much spinning] (performance)

→ Fairness (Starvation)



As contention increases....



Bad case: interrupted in critical section ⇒ WASTE

100 threads (not 2)

⇒ T_1 (inter.)

$\underbrace{t_2}$ ... T_{100}

for time slice (10 ms)

$\sim \underline{100} \times \underline{10\,ms} \Rightarrow \boxed{1\,s}$

length of time slice

(assuming ~100 threads)

Fairness : (100 x cost of context switch)

T_1

$\vdots$ contending for lock

T_N

T_i is spinning ;
will it ever acquire lock ?
[who knows?]

Fairness / Starvation via

<u>Ticket</u> <u>Lock</u>:

```
ticket = 0;
turn = 0;

acquire (    ) {
    my_turn = fetch();
```

$T_1 \rightarrow$ acquire
my_turn = 0;

$T_2 \rightarrow$ acquire
my_turn = 1;

T_1: it's my turn : (0)

acquire lock

T_2: won't acquire lock
(1 != 0)

T_1: release : allows
inc turn T_2 to
get lock

H/W primitives:

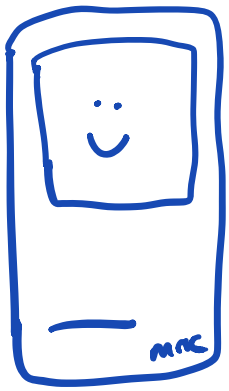
xchg, fetch-and-add

OS support :

scheduler might
want to know
process is spinning
mindlessly

→ yield() system call

⇒ Running process\thread calls yield → RUNNING ⇒ RUNNABLE



needed yield() (BAD)

Solaris: early multi-threaded

OS primitive:

⇒ park() → like yield but ⇒ BLOCKED

⇒ unpark(t_id) make thread t_id RUNNABLE

(not RUNNABLE)

[Piazza]

~~537-help@cs~~

good for you

me

~~bad for TAs~~

Grades:

summary: coming

Scores $P_{1A} \ldots P_{2B}$

approx $\Rightarrow$ letter

A, B, ...

Contests: 1A, 2A } [done soon]

$\Rightarrow$ T-shirts, etc.

<u>Exam</u>: (more on this next week)