

CPSC 457 T03/T04

Week 4 Day 1

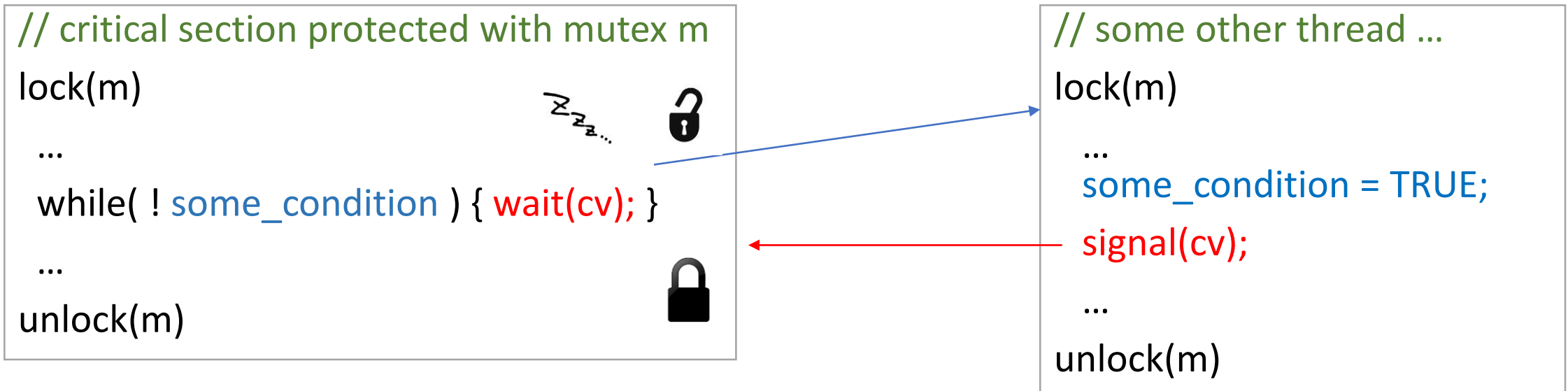
Xining Chen

Agenda

- Condition variable
- unique_lock
- lock_guard
- Semaphores
- Assignment 3

Condition variables

- A synchronization primitive
- Used together with mutexes
- *When to use:* critical sections w/ **loops waiting** for some **'condition'** to happen
- This **'condition'** can only become true if **another thread** runs its critical section



Download CPSC 457 – Synchronization – Code.pdf from D2L under W4D1

- Exercise + Demo

CV in C++

unique lock in C++

lock_guard in C++

Semaphores

- Semaphores can be used to provide **mutual exclusion**.
- A semaphore has **integer value**.
- A locked semaphore can be unlocked by any thread
- Three operations: **initialization**, **decrement** (wait), **increment** (signal)
- decrement and increment must be executed **atomically**

Mutex	Semaphore
Binary operation/value (lock/unlock)	Integer value
Lock/unlock done by the same thread	Locked threads can be unlocked by any thread

Binary semaphore

- Behave very similarly to mutex locks.
- On systems that do not support mutex locks, binary semaphores can be used instead to provide mutual exclusion.
- Note: A locked thread can be unlocked by any thread using a semaphore.

Counting semaphores

- Can be used to control access to some given resource consisting of a finite number of instances.
 - **Initialize** the semaphore to the **number of resources available**
 - Each process that uses a resource performs a **wait()** operation (aka. **decrement** resource count)
 - When the count for the semaphore goes to 0, all resources are being used.
 - Additional processes that wish to use a resource will now be blocked until count > 0.

Assignment 3

Ignore 0,1

<u>Input:</u>	25	4012009	165
<u>Factors:</u>	1, 5	1, 2003, 4012009 * Typo on A3	1, 3, 5, ..., 165

Ignore 0,1

<u>Input:</u>	25	4012009	165
<u>Factors:</u>	1, 5	1, 2003, 4012009 * Typo on A3	1, 3, 5, ..., 165
Smallest non-trivial factor :	5	2003	3

Ignore 0,1

<u>Input:</u>	25	4012009	165
<u>Factors:</u>	1, 5	1, 2003, 4012009 * Typo on A3	1, 3, 5, ..., 165
Smallest non-trivial factor :	5	2003	3

$$\Sigma = 5 + 2003 + 3 = 2011$$

Output

Solution 1 : Let $N=2$

5	}	Thread #1
10		
33		
1123		
56	}	Thread #2
3		
10000		

Solution 1 : Let $N=2$

5	}	Thread #1
10		
33		
1123		
56	}	Thread #2
3		
10000		

Not Good.

Ex://

	[132456799817	
#1	[10	↑
		33	
		1123	
#2	[56	
		3	
		100	

Thread #1

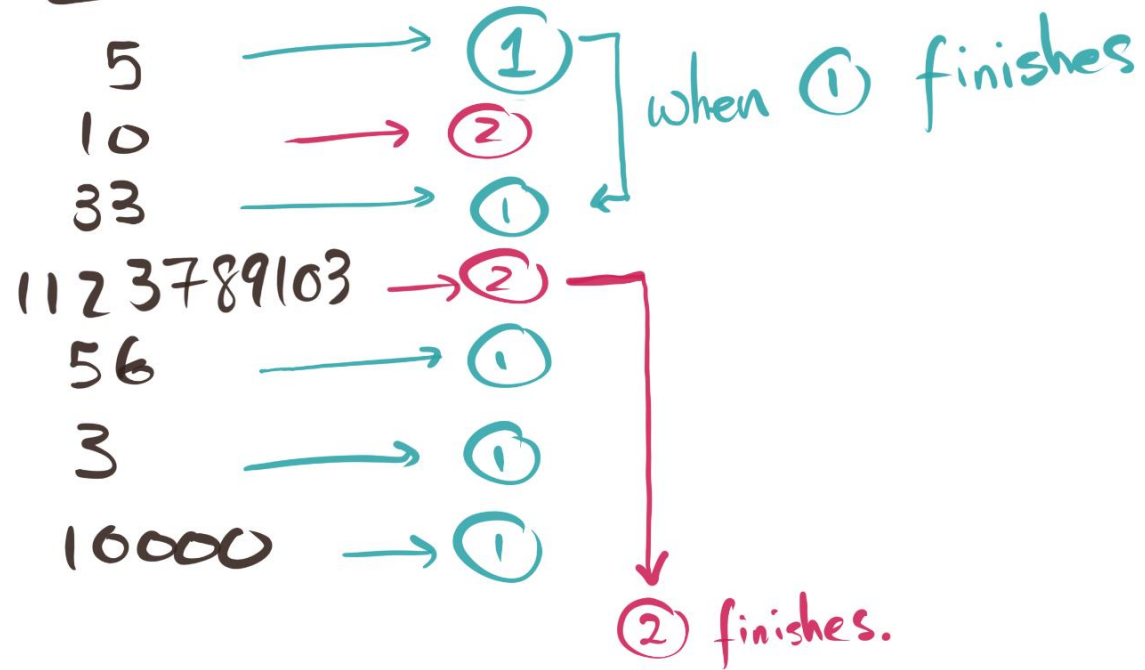
bottleneck.

⇒ runtime will
be similar to
single thread code.

Solution #2 : Let $N=2$



Solution #2 : Let $N=2$



OK solution.

Runtime?

However, consider input: 1123789103.

Solution #3: Let $N=2$

5 \rightarrow ① ②

10 \rightarrow ① ②

33
1123789103 \rightarrow ① ②

56 \rightarrow ① ②

3
10000 \rightarrow ① ②

Parallelize getSmallestDivisor().

Assignment 3 suggestions

1. Implement solution #2 first.
2. Use mutex, condition variable, semaphores, and pthread_barrier to try and implement solution #3.

Next Time

- pthread versions of everything
- Assignment 3 help (?)