Little Book of Semaphores, Chapter 1

Geoffrey Matthews Western Washington University

January 9, 2015

Synchronization

- When more than one thread is running, synchronization may be important:
- Serialization: Event A must happen before event B.
- Mutual exclusion: Events A and B must not happen at the same time.

Threads

- For a single core, there is only one instruction happening at a time.
- A sequence of such instructions is a thread.
- A desktop computer will have many threads running at one time.
- The OS may run threads in parallel, on different CPUs.
- The OS may run only one thread at a time, and interleave many threads on a single CPU.

Lunch

- You and your friend Bob live in different cities.
- You wonder who ate lunch first.
- How can you find out?

Lunch

- You and your friend Bob live in different cities.
- You wonder who ate lunch first.
- How can you find out?
- You could call and ask the time, but how would you know if your clocks were synchronized?

- You want to guarantee that you ate lunch before Bob.
- How can you do it?

- You want to guarantee that you ate lunch before Bob.
- How can you do it?
- Synchronize with messages:

You ______ You _____ Eat breakfast Work Eat lunch Call Bob

Eat breakfast
Wait for call
Eat lunch

- You want to guarantee that you ate lunch before Bob.
- How can you do it?
- Synchronize with messages:

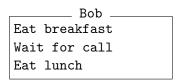
You

Eat breakfast

Work

Eat lunch

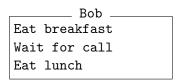
Call Bob



- You ate lunch sequentially (order guaranteed)
- You ate breakfast concurrently (order undetermined)

- You want to guarantee that you ate lunch before Bob.
- How can you do it?
- Synchronize with messages:

You ______ You _____ Eat breakfast Work Eat lunch Call Bob



- You ate lunch sequentially (order guaranteed)
- You ate breakfast concurrently (order undetermined)
- Two events are concurrent if we cannot tell by looking at the program which will happen first.

- You and Bob work in the same office, but are NOT talking to each other.
- How can you guarantee you don't have lunch at the same time?

- You and Bob work in the same office, but are NOT talking to each other.
- How can you guarantee you don't have lunch at the same time?
- Keep a rock in a box.

- You and Bob work in the same office, but are NOT talking to each other.
- How can you guarantee you don't have lunch at the same time?
- Keep a rock in a box.
- Make a rule that you can only eat lunch if you have the rock.

- You and Bob work in the same office, but are NOT talking to each other.
- How can you guarantee you don't have lunch at the same time?
- Keep a rock in a box.
- Make a rule that you can only eat lunch if you have the rock.

 Bob ______

 Bob _____

Eat breakfast
Take rock from box
Eat lunch
Put rock back in box

- You and Bob work in the same office, but are NOT talking to each other.
- How can you guarantee you don't have lunch at the same time?
- Keep a rock in a box.
- Make a rule that you can only eat lunch if you have the rock.

 Bob ______

 Bob _____

Eat breakfast
Take rock from box
Eat lunch
Put rock back in box

Eat breakfast
Take rock from box
Eat lunch
Put rock back in box

You ate breakfast concurrently (order undetermined). Lunch?

- You and Bob work in the same office, but are NOT talking to each other.
- How can you guarantee you don't have lunch at the same time?
- Keep a rock in a box.

Eat breakfast
Take rock from box
Eat lunch
Put rock back in box

- You ate breakfast concurrently (order undetermined). Lunch?
- You ate lunch concurrently (order undetermined).

- You and Bob work in the same office, but are NOT talking to each other.
- How can you guarantee you don't have lunch at the same time?
- Keep a rock in a box.
- Make a rule that you can only eat lunch if you have the rock.

 Bob ______

 Bob _____

Eat breakfast
Take rock from box
Eat lunch
Put rock back in box

- You ate breakfast concurrently (order undetermined). Lunch?
- You ate lunch concurrently (order undetermined).
- This version does not enforce order, but **mutual exclusion**.

- You and Bob work in the same office, but are NOT talking to each other.
- How can you guarantee you don't have lunch at the same time?
- Keep a rock in a box.
- Make a rule that you can only eat lunch if you have the rock.

 Bob ______

 Bob _____

Eat breakfast
Take rock from box
Eat lunch
Put rock back in box

- You ate breakfast concurrently (order undetermined). Lunch?
- You ate lunch concurrently (order undetermined).
- This version does not enforce order, but mutual exclusion.
- Can you find a version that enforces order?

- You and Bob work in the same office, but are NOT talking to each other.
- How can you guarantee you don't have lunch at the same time?
- Keep a rock in a box.
- Make a rule that you can only eat lunch if you have the rock.

 Bob ______

 Bob _____

Eat breakfast
Take rock from box
Eat lunch
Put rock back in box

- You ate breakfast concurrently (order undetermined). Lunch?
- You ate lunch concurrently (order undetermined).
- This version does not enforce order, but mutual exclusion.
- Can you find a version that enforces order?
- Can you find a message version that enforces mutual exclusion?



x = 5
print x

```
x = 7 Thread B
```

- What path yields output 5 and final value 5?
- What path yields output 7 and final value 7?

Thread A _______ x = 5 print x

```
x = 7 Thread B
```

- What path yields output 5 and final value 5?
- What path yields output 7 and final value 7?
- What path yields output 5 and final value 7?

x = 5
print x

```
x = 7 Thread B
```

- What path yields output 5 and final value 5?
- What path yields output 7 and final value 7?
- What path yields output 5 and final value 7?
- What path yields output 7 and final value 5?

```
x = 5
print x
```

```
x = 7 Thread B
```

- What path yields output 5 and final value 5?
- What path yields output 7 and final value 7?
- What path yields output 5 and final value 7?
- What path yields output 7 and final value 5?
- What paths are possible and what are their effects?
- Can we prove bad effects impossible, and desirable effects certain?

Concurrent Updates

Concurrent Updates

$$x = x + 1$$

Thread A load x add 1 store x

Thread B load x add 1 store x

Concurrent Updates

Any operation that cannot be interrupted is said to be atomic.