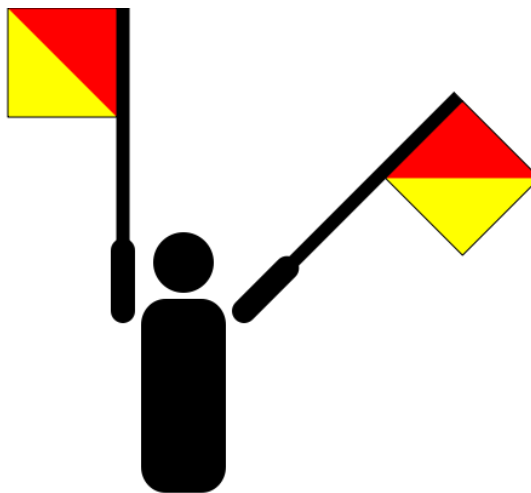


Too Few Chairs

Threads A, B, C, ...

```
1  if (chairs > 0) {  
2      chairs--;  
3      print("I'm working!");  
4      chairs++;  
5  }
```

Semaphores



Too Much Milk with Semaphores

```
sem = new Semaphore(1);
```

Threads A, B

```
1 sem.wait();  
2 if (noMilk) {  
3     buy milk;  
4 }  
5 sem.signal();
```

Too Few Chairs with Semaphores

```
sem = new Semaphore(NUM_CHAIRS);
```

Threads A, B, C, ...

```
1 sem.wait();  
2 print("I'm working!");  
3 sem.signal();
```

Implementing Semaphores

```
class Semaphore {
public:
    void Wait(Thread T);
    void Signal();

private:
    int value;
    Queue Q;
}

Semaphore(int val) {
    value = val;
    Q = empty;
}

Wait() {
    value--;
    if (value < 0) {
        add curThread to Q;
        put curThread to sleep;
    }
}

Signal() {
    value++;
    if (value <= 0){
        remove T from Q;
        put T on ready queue;
    }
}
```

Must be atomic (interrupts or Test&Set)!

Producers/Consumers with Semaphores

```
class ProducerConsumer {
public:
    void Produce();
    void Consume();
private:
    Items buffer;
    // control buffer access
    Semaphore mutex;
    // count of free slots
    Semaphore empty;
    // count of used slots
    Semaphore full;
}

ProducerConsumer(int N) {
    mutex.value = 1;
    empty.value = N;
    full.value = 0;
    buffer = new buffer[N];
}

void Produce() {
    <produce item>
    empty.Wait(); // one fewer slot, or wait
    mutex.Wait(); // get access to buffers
    <add item to buffer>
    mutex.Signal(); // release buffers
    full.Signal(); // one more used slot
}

void Consume() {
    full.Wait(); // wait until there's an item
    mutex.Wait(); // get access to buffers
    <remove item from buffer>
    mutex.Signal(); // release buffers
    empty.Signal(); // one more free slot
    <use item>
}
```

Locks and Semaphores Summary

- Synchronization difficult without OS help
- Primitives: Locks, Semaphores
- Disable interrupts or test&set
 - Hardware support!
- Semaphores generalize locks
 - Binary semaphores
 - Counting semaphores
 - Scheduling constraints