

# CS 472 – Provably Correct Programming

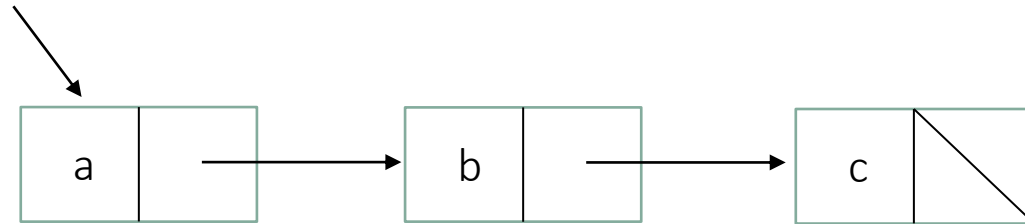
William Mansky

## Questions

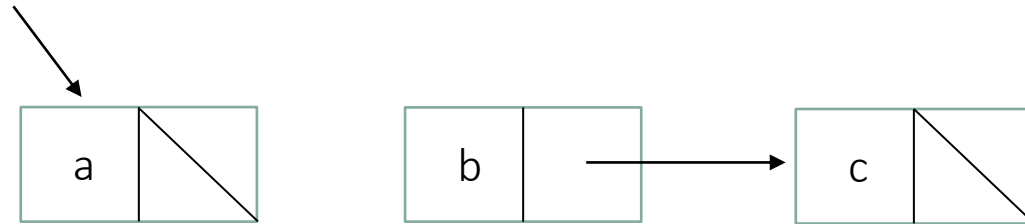
Nobody has responded yet.

Hang tight! Responses are coming in.

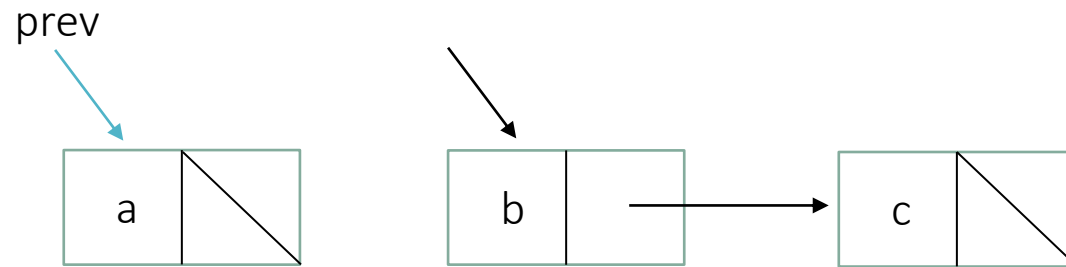
# List Reverse



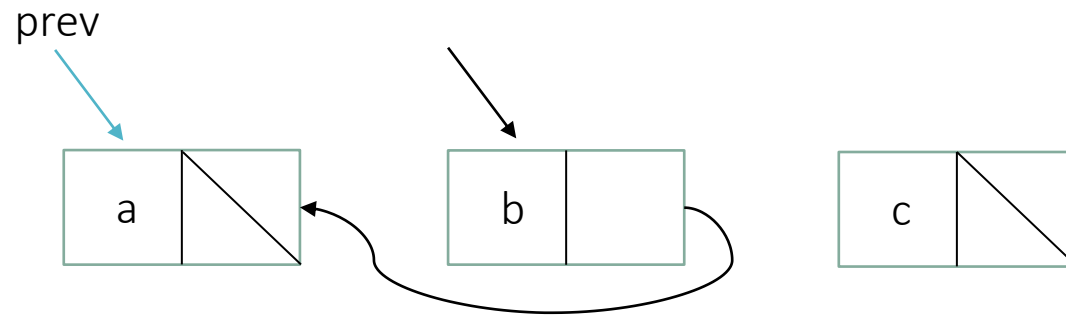
# List Reverse



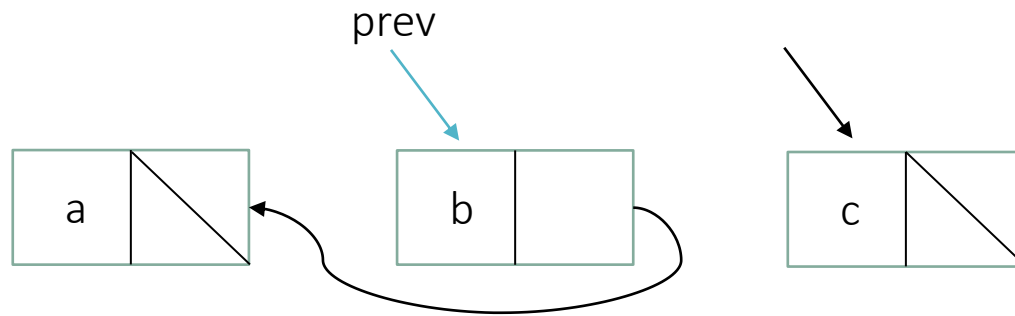
# List Reverse



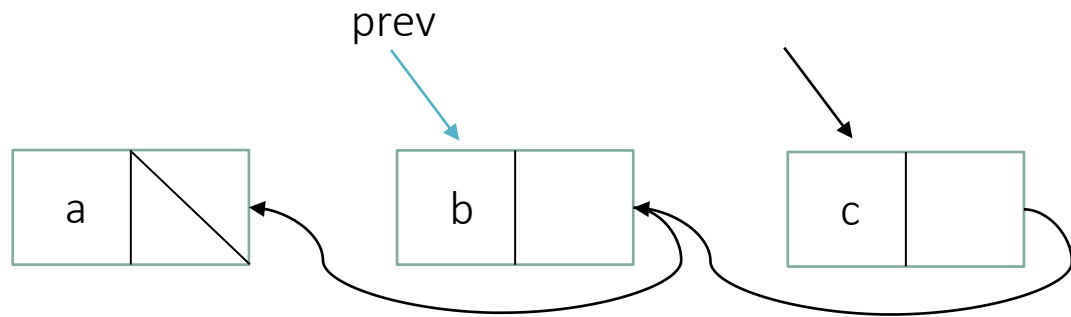
# List Reverse



# List Reverse



# List Reverse



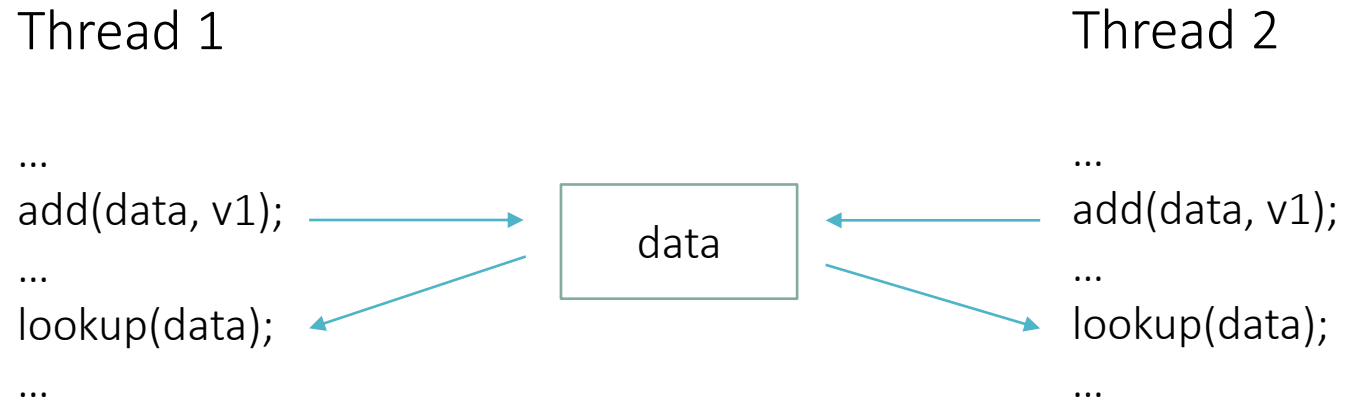


## Questions

Nobody has responded yet.  
Hang tight! Responses are coming in.

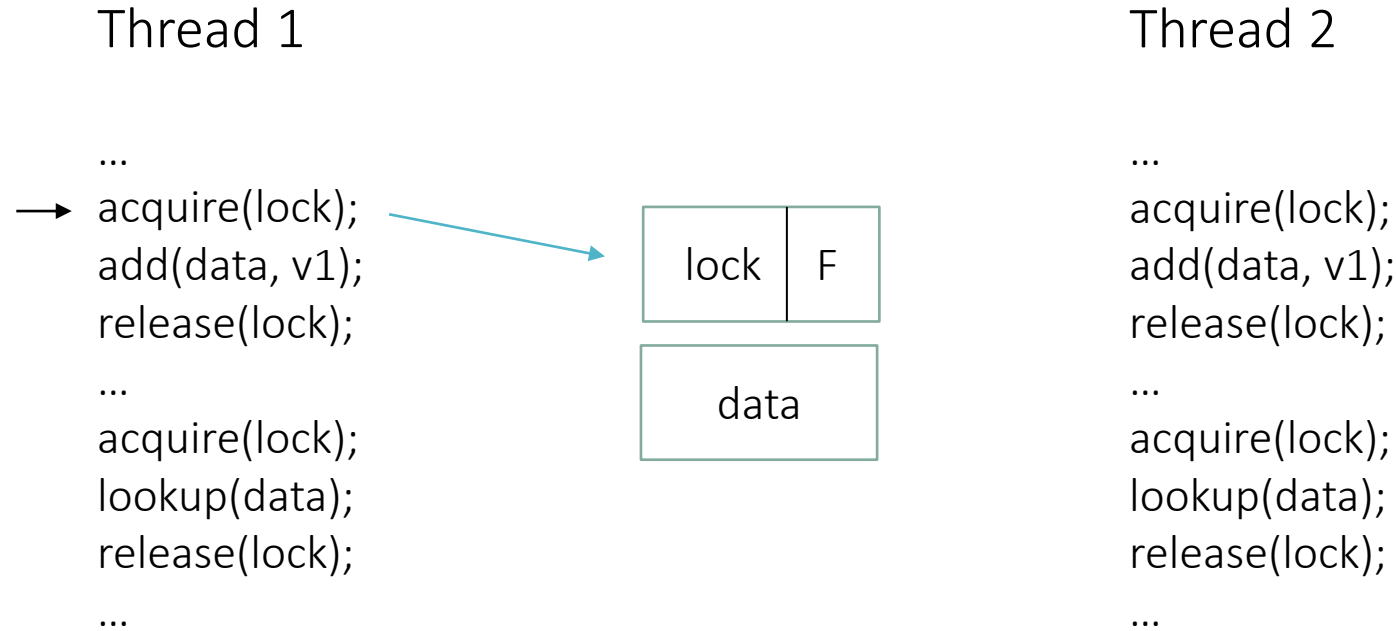


# Mutual Exclusion



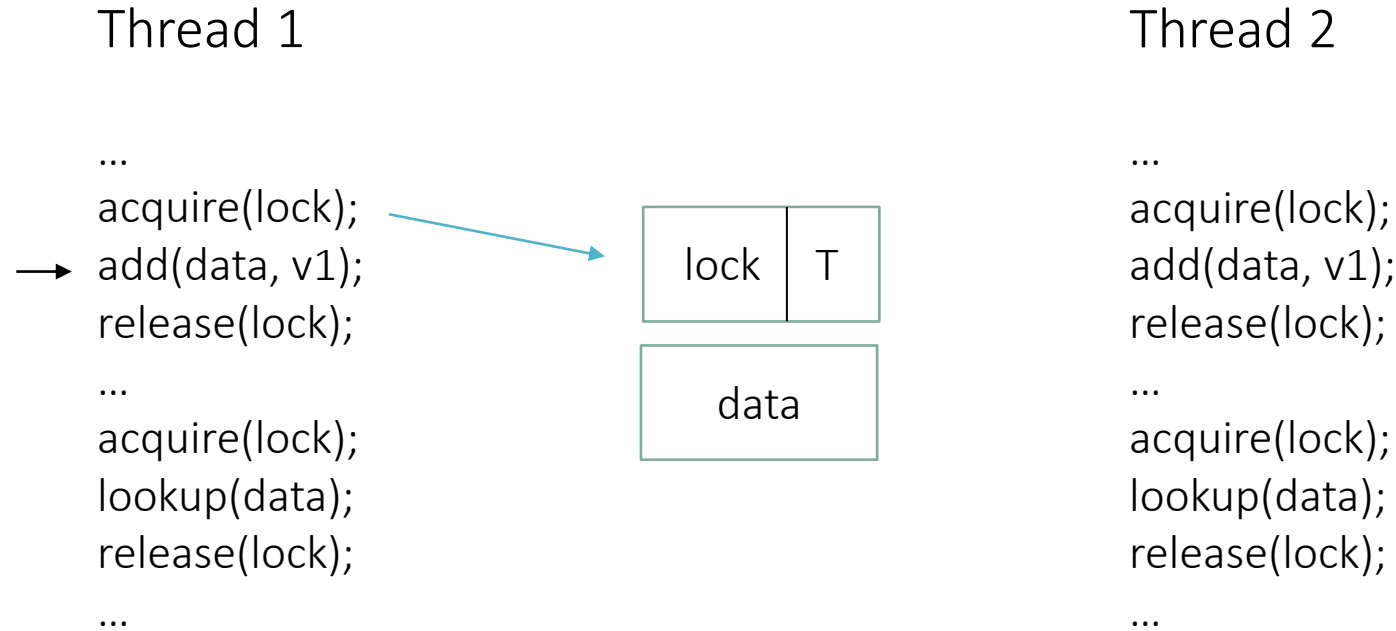
- If both threads try to add at the same time, something might go wrong

# Mutual Exclusion



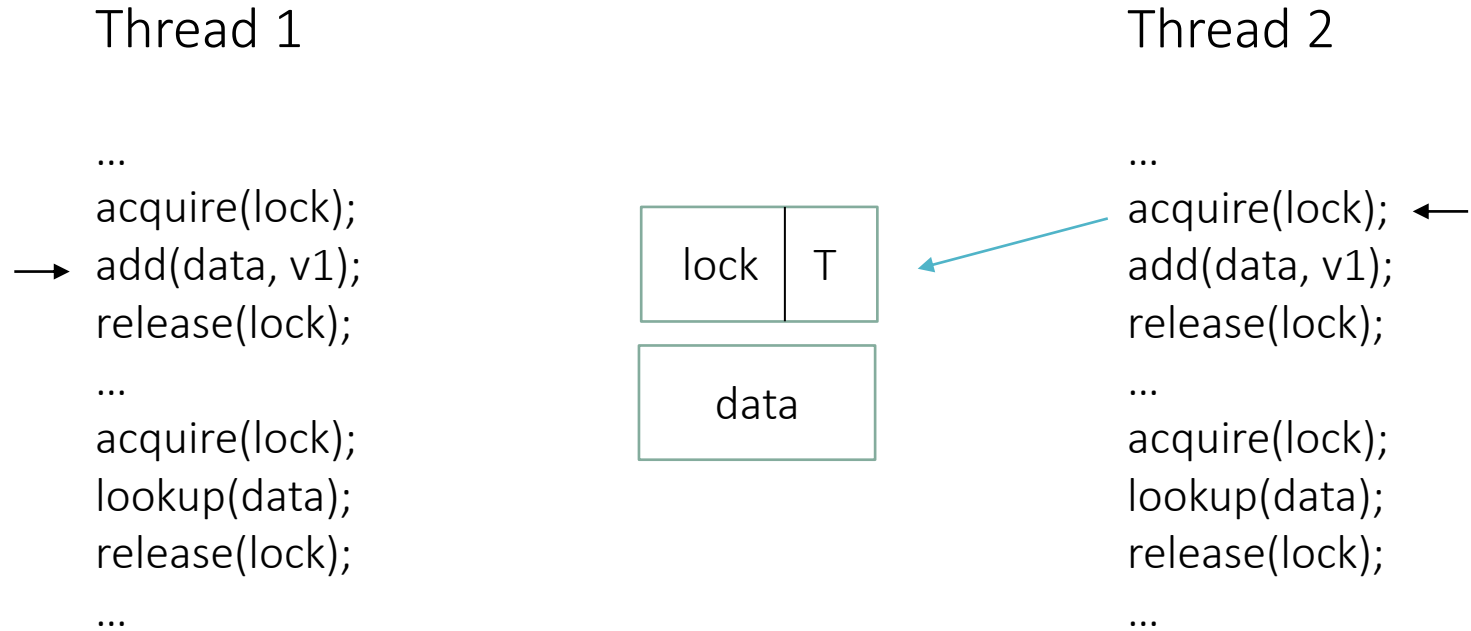
- If both threads try to add at the same time, something might go wrong

# Mutual Exclusion



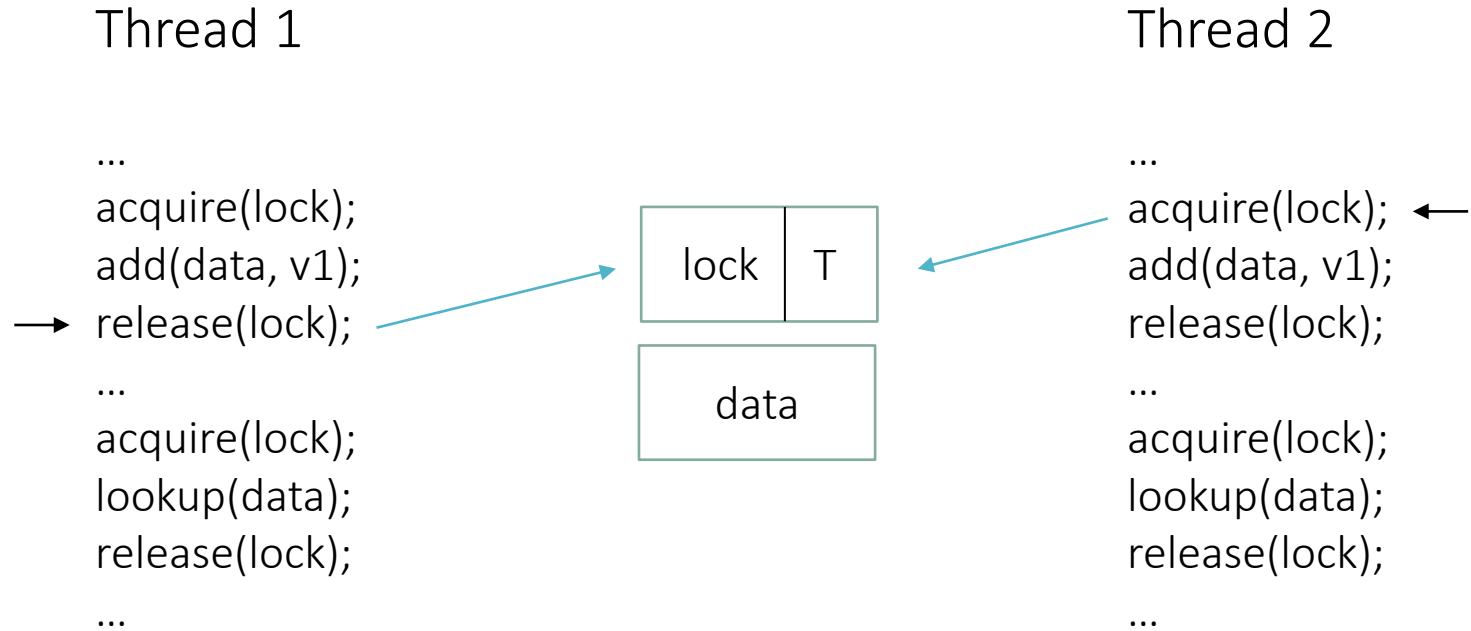
- If both threads try to add at the same time, something might go wrong

# Mutual Exclusion



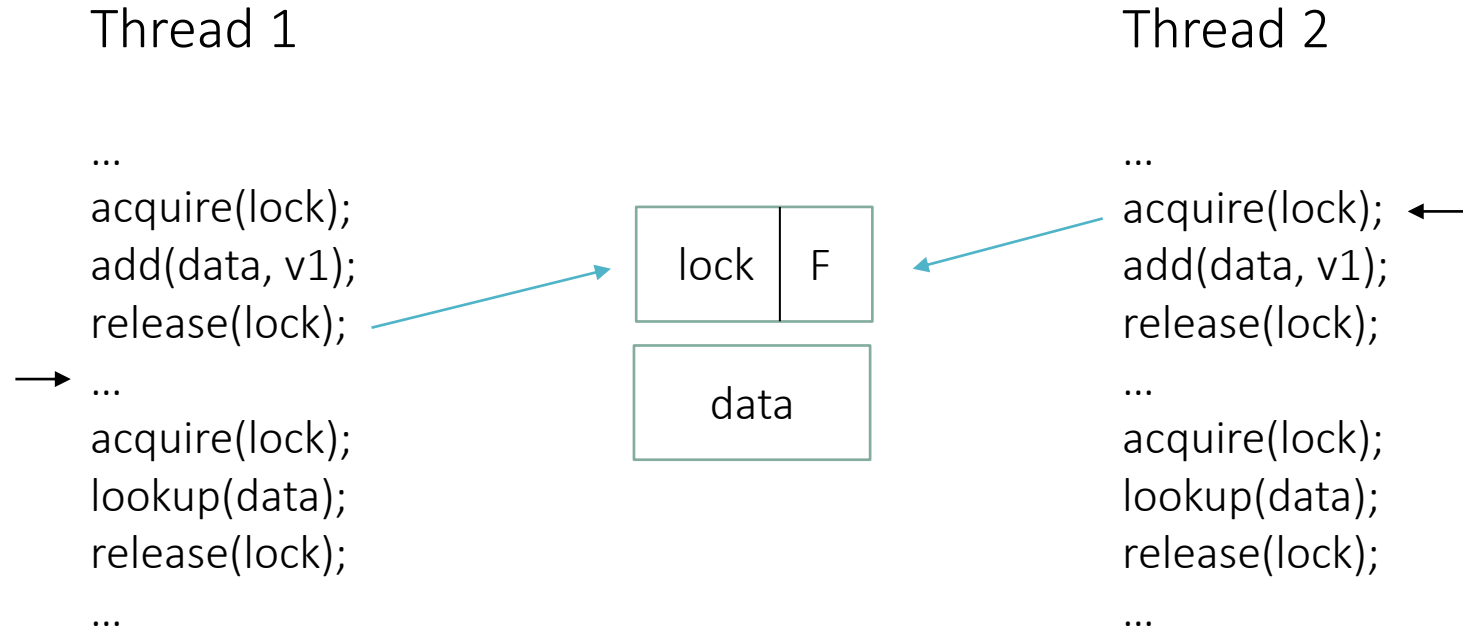
- If both threads try to add at the same time, something might go wrong

# Mutual Exclusion



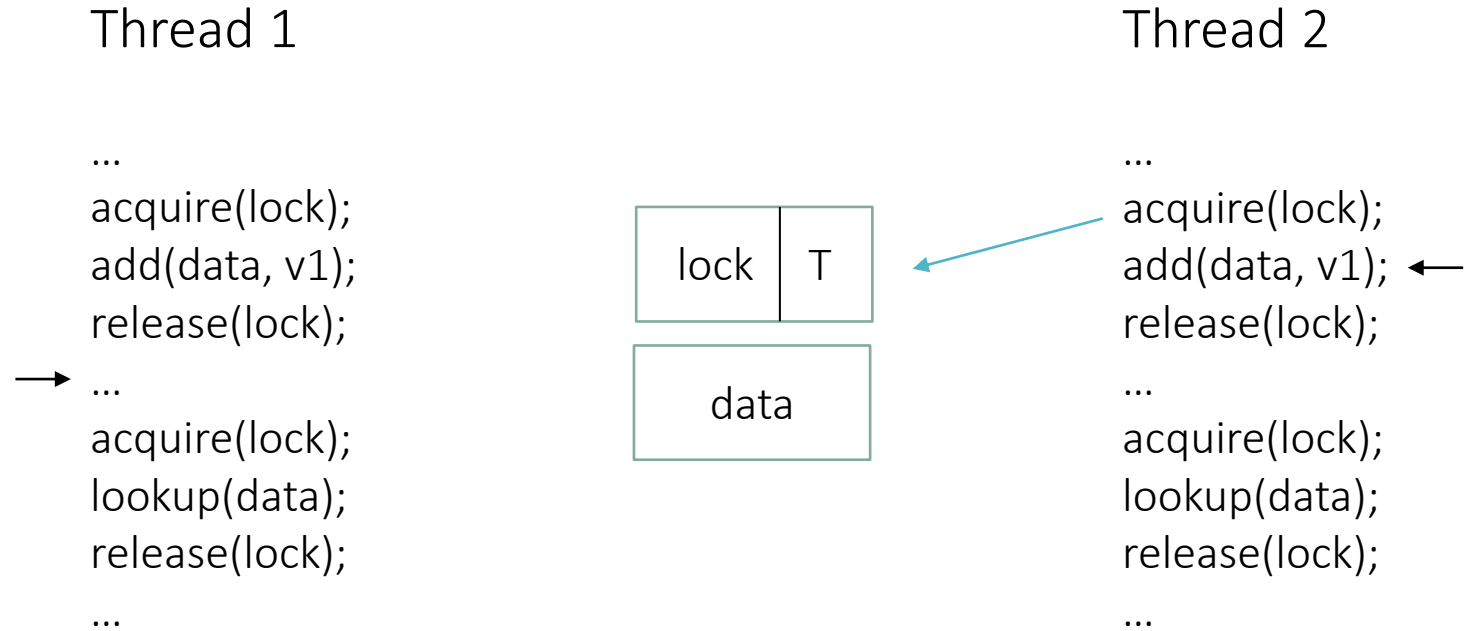
- If both threads try to add at the same time, something might go wrong

# Mutual Exclusion



- If both threads try to add at the same time, something might go wrong

# Mutual Exclusion



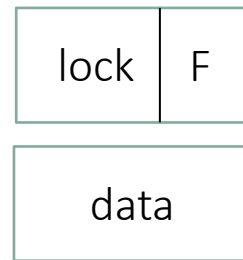
- If both threads try to add at the same time, something might go wrong
- Key property: only one thread holds the lock at a time
  - And so only one thread accesses the data at a time



# Mutual Exclusion: Acquire

Thread 1

```
...  
acquire(lock);  
add(data, v1);  
release(lock);  
...  
acquire(lock);  
lookup(data);  
release(lock);  
...
```



Thread 2

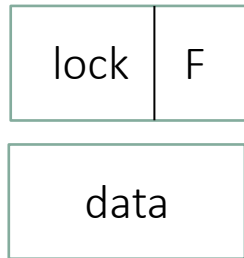
```
...  
acquire(lock);  
add(data, v1);  
release(lock);  
...  
acquire(lock);  
lookup(data);  
release(lock);  
...
```

- Key property: only one thread holds the lock at a time

# Mutual Exclusion: Acquire

Thread 1

```
...  
acquire(lock);  
add(data, v1);  
release(lock);  
...  
acquire(lock);  
lookup(data);  
release(lock);  
...
```



Thread 2

```
...  
acquire(lock);  
add(data, v1);  
release(lock);  
...  
acquire(lock);  
lookup(data);  
release(lock);  
...
```

acquire:

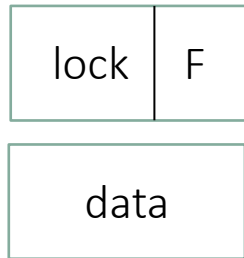
```
thread 1 =>  
thread 2 => if lock is F  
              set it to T  
              else wait
```

- Key property: only one thread holds the lock at a time

# Mutual Exclusion: Acquire

Thread 1

```
...  
acquire(lock);  
add(data, v1);  
release(lock);  
...  
acquire(lock);  
lookup(data);  
release(lock);  
...
```



Thread 2

```
...  
acquire(lock);  
add(data, v1);  
release(lock);  
...  
acquire(lock);  
lookup(data);  
release(lock);  
...
```

acquire:

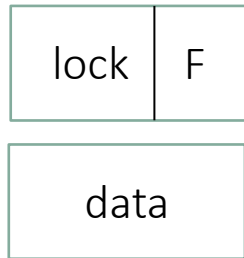
```
thread 2 → if lock is F  
thread 1 → set it to T  
else wait
```

- Key property: only one thread holds the lock at a time

# Mutual Exclusion: Acquire

Thread 1

```
...  
acquire(lock);  
add(data, v1);  
release(lock);  
...  
acquire(lock);  
lookup(data);  
release(lock);  
...
```



Thread 2

```
...  
acquire(lock);  
add(data, v1);  
release(lock);  
...  
acquire(lock);  
lookup(data);  
release(lock);  
...
```

acquire:

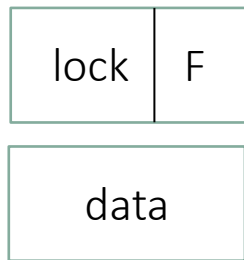
```
thread 1 →  
thread 2 →  
if lock is F  
  set it to T ⚠  
else wait
```

- Key property: only one thread holds the lock at a time

# Mutual Exclusion: Acquire

Thread 1

```
...  
acquire(lock);  
add(data, v1);  
release(lock);  
...  
acquire(lock);  
lookup(data);  
release(lock);  
...
```



Thread 2

```
...  
acquire(lock);  
add(data, v1);  
release(lock);  
...  
acquire(lock);  
lookup(data);  
release(lock);  
...
```

acquire:

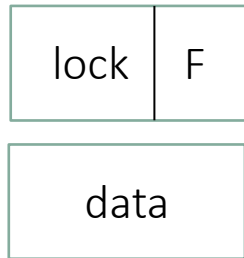
thread 1  $\Rightarrow$  if lock is F set it to T  
thread 2  $\Rightarrow$  else wait

- Key property: only one thread holds the lock at a time

# Mutual Exclusion: Acquire

Thread 1

```
...  
acquire(lock);  
add(data, v1);  
release(lock);  
...  
acquire(lock);  
lookup(data);  
release(lock);  
...
```



Thread 2

```
...  
acquire(lock);  
add(data, v1);  
release(lock);  
...  
acquire(lock);  
lookup(data);  
release(lock);  
...
```

acquire:

```
thread 1 ⇒ CAS(lock, F, T)  
thread 2 ⇒ else wait
```

- Key property: only one thread holds the lock at a time
- CAS (compare-and-set) operation lets us check and set in one step

## Questions

Nobody has responded yet.

Hang tight! Responses are coming in.