


Lec-31

Leetcode concurrency

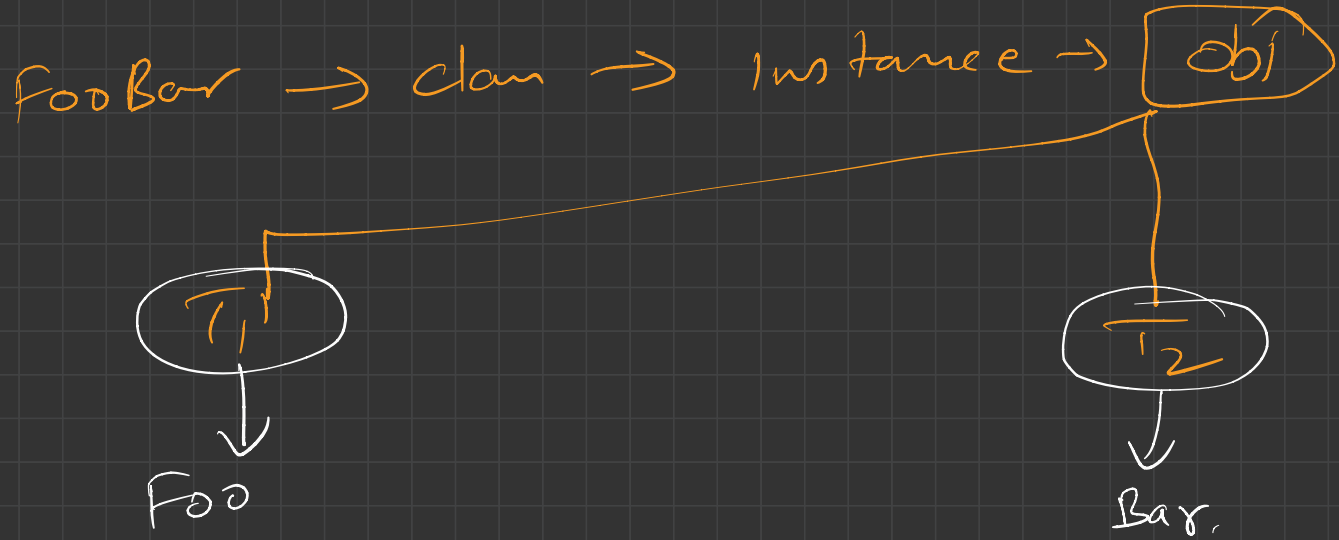
③

① Print FooBar

② Print zero even odd

③ Building 1120 ~~★~~

① Print → FooBar



Single Process/thread → Foo Bar Foo Bar.

Two threads → $\begin{matrix} T_1 \rightarrow \\ T_2 \rightarrow \end{matrix}$ $\begin{matrix} T_1 \rightarrow \\ T_2 \rightarrow \end{matrix}$

Code

Watch

Lec-23

```
1  class FooBar {
2  private:
3      int n;
4      std::mutex m;
5      std::condition_variable cv;
6      bool turn;
7  public:
8      FooBar(int n) {
9          this->n = n;
10         turn = 0;
11     }
12
13     void foo(function<void()> printFoo) {
14
15         for (int i = 0; i < n; i++) {
16             std::unique_lock<std::mutex> lock(m);
17             while(turn == 1){
18                 cv.wait(lock);
19             }
20             printFoo();
21             turn = 1;
22             cv.notify_all();
23         }
24     }
25
26     void bar(function<void()> printBar) {
27
28         for (int i = 0; i < n; i++) {
29             std::unique_lock<std::mutex> lock(m);
30             while(turn == 0){
31                 cv.wait(lock);
32             }
33             printBar();
34             turn = 0;
35             cv.notify_all();
36         }
37     }
38 };
```

Initially

turn = 0



T₁



turn = 1



T₂

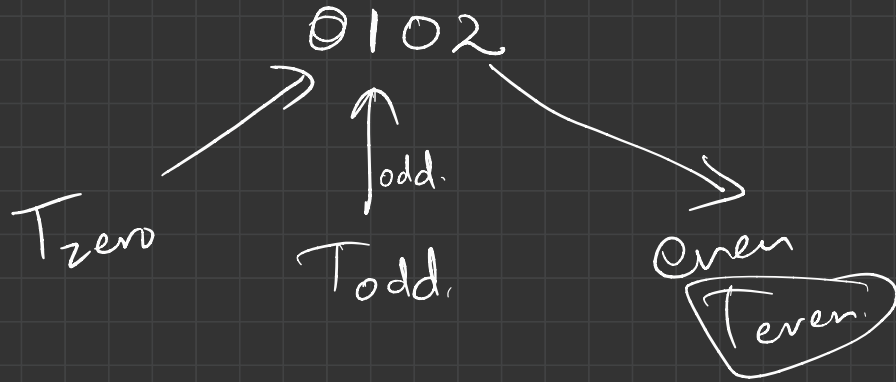


turn = 0

Problem 2

Print zero even odd

⇒ $n = 2$



Solution 1

$n=3 \rightarrow 010203$
↑ ↑ ↑
Turn Turn Turn

$n=2$

```
1 class ZeroEvenOdd {
2 private:
3     int n;
4     std::mutex m;
5     std::condition_variable cv;
6     int turn;
7     int i;
8 public:
9     ZeroEvenOdd(int n) {
10         this->n = n;
11         turn = 0;
12         i = 1;
13     }
14
15     // printNumber(x) outputs "x", where x is an integer
16     void zero(function<void(int)> printNumber) {
17         while(i <= n){
18             std::unique_lock<std::mutex> lock(m);
19             while(turn != 0 && i <= n){
20                 cv.wait(lock);
21             }
22             if(i > n){
23                 break;
24             }
25             printNumber(0);
26             turn = (i % 2) == 0 ? 2 : 1;
27             cv.notify_all();
28         }
29     }
30 }
```

Turn = 0

```
31 void even(function<void(int)> printNumber) {
32     while(i <= n){
33         std::unique_lock<std::mutex> lock(m);
34         while(turn != 2 && i <= n){
35             cv.wait(lock);
36         }
37         if(i > n){
38             break;
39         }
40         printNumber(i++);
41         turn = 0;
42         cv.notify_all();
43     }
44 }
45
46 void odd(function<void(int)> printNumber) {
47     while(i <= n){
48         std::unique_lock<std::mutex> lock(m);
49         while(turn != 1 && i <= n){
50             cv.wait(lock);
51         }
52         if(i > n){
53             break;
54         }
55         printNumber(i++);
56         turn = 0;
57         cv.notify_all();
58     }
59 }
60 }
```

$i=3$ turn = 2

turn = 1

Case 1, $n=2$

- ① $i=1 \rightarrow \text{turn} = 1$
- ② $i=2 \rightarrow \text{turn} = 2$
- ③ $i=3$

output 0102
↓
✓✓

③ n20

2)

n → Threads → multiple

0 → Threads → multiple

[nno] [non]

input → [00|nnnn]

↑ ↑ ↑ ↑

↓

2 0 threads

→ 4 n threads

Solution

```
1 class H2O {
2     std::mutex m;
3     std::condition_variable cv;
4     int turn;
5 public:
6     H2O() {
7         turn = 0;
8     }
9
10    void hydrogen(function<void()> releaseHydrogen) {
11        std::unique_lock<std::mutex> lock(m);
12        while(turn == 2){
13            cv.wait(lock);
14        }
15        releaseHydrogen();
16        ++turn;
17        cv.notify_all();
18    }
19
20    void oxygen(function<void()> releaseOxygen) {
21        std::unique_lock<std::mutex> lock(m);
22        while(turn < 2){
23            cv.wait(lock);
24        }
25        releaseOxygen();
26        turn = 0;
27        cv.notify_all();
28    }
29};
```

turn = 0
↓ H
turn = 1
↓ H
turn = 2
↓ O
turn = 0

Conclusion |

turn →

mutex

Conditions