# 题目

假设有这么一个类：

class ZeroEvenOdd {

  public ZeroEvenOdd(int n) { ... }  // 构造函数

public void zero(printNumber) { ... } // 仅打印出 0

public void even(printNumber) { ... } // 仅打印出 偶数

public void odd(printNumber) { ... } // 仅打印出 奇数

}

相同的一个 ZeroEvenOdd 类实例将会传递给三个不同的线程：

线程 A 将调用 zero()，它只输出 0 。

线程 B 将调用 even()，它只输出偶数。

线程 C 将调用 odd()，它只输出奇数。

每个线程都有一个 printNumber 方法来输出一个整数。请修改给出的代码以输出整数序列 010203040506... ，其中序列的长度必须为 2n。

示例 1：

输入：n = 2

输出："0102"

说明：三条线程异步执行，其中一个调用 zero()，另一个线程调用 even()，最后一个线程调用odd()。正确的输出为 "0102"。

示例 2：

输入：n = 5

输出："0102030405"

# 分析

## 方法一：互斥锁

class ZeroEvenOdd {

private:

int n;

pthread\_mutex\_t mutex0;

pthread\_mutex\_t mutex1;

pthread\_mutex\_t mutex2;

public:

ZeroEvenOdd(int n) {

this->n = n;

pthread\_mutex\_init(&mutex0, NULL);

pthread\_mutex\_init(&mutex1, NULL);

pthread\_mutex\_init(&mutex2, NULL);

//lock

pthread\_mutex\_lock(&mutex1);

pthread\_mutex\_lock(&mutex2);

}

// printNumber(x) outputs "x", where x is an integer.

void zero(function<void(int)> printNumber) {

for(int i = 1; i <= n; i++) {

pthread\_mutex\_lock(&mutex0);

printNumber(0);

if(i & 1) pthread\_mutex\_unlock(&mutex1);

else pthread\_mutex\_unlock(&mutex2);

}

}

void even(function<void(int)> printNumber) {

for(int i = 2; i <= n; i+=2) {

pthread\_mutex\_lock(&mutex2);

printNumber(i);

pthread\_mutex\_unlock(&mutex0);

}

}

void odd(function<void(int)> printNumber) {

for(int i = 1; i <= n; i+=2) {

pthread\_mutex\_lock(&mutex1);

printNumber(i);

pthread\_mutex\_unlock(&mutex0);

}

}

};

## 方法二：互斥锁+条件变量

class ZeroEvenOdd {

private:

int n;

std::condition\_variable g\_cv;

std::mutex mt;

bool ze,other;

public:

ZeroEvenOdd(int n) : g\_cv(), mt(), ze(false), other(true){

this->n = n;

}

// printNumber(x) outputs "x", where x is an integer.

void zero(function<void(int)> printNumber) {

for(int i = 1; i <= n; ++i){

std::unique\_lock<std::mutex> lk(mt);

g\_cv.wait(lk, [=](){return !ze && other;});

printNumber(0);

if(i&1)ze = true;

else other = false;

g\_cv.notify\_all();

}

}

void even(function<void(int)> printNumber) {

for(int i = 2; i <= n; i+=2){

std::unique\_lock<std::mutex> lk(mt);

g\_cv.wait(lk,[=](){return !ze && !other;});

printNumber(i);

ze = false;

other = true;

g\_cv.notify\_all();

}

}

void odd(function<void(int)> printNumber) {

for(int i = 1; i <= n; i+=2){

std::unique\_lock<std::mutex> lk(mt);

g\_cv.wait(lk, [=](){return ze && other;});

printNumber(i);

ze = false;

g\_cv.notify\_all();

}

}

};

## 方法三：信号量

#include <semaphore.h>

class ZeroEvenOdd {

private:

int n;

sem\_t zero\_sem;

sem\_t odd\_sem;

sem\_t even\_sem;

public:

ZeroEvenOdd(int n) {

this->n = n;

sem\_init(&zero\_sem, 0, 1); //初始化一个0

sem\_init(&odd\_sem, 0, 0);

sem\_init(&even\_sem, 0, 0);

}

// printNumber(x) outputs "x", where x is an integer.

void zero(function<void(int)> printNumber) {

for(int i = 1; i <= n; i++) {

sem\_wait(&zero\_sem);

printNumber(0);

if(i & 1) sem\_post(&odd\_sem);

else sem\_post(&even\_sem);

}

}

void even(function<void(int)> printNumber) {

for(int i = 2; i <= n; i+=2) {

sem\_wait(&even\_sem);

printNumber(i);

sem\_post(&zero\_sem);

}

}

void odd(function<void(int)> printNumber) {

for(int i = 1; i <= n; i+=2) {

sem\_wait(&odd\_sem);

printNumber(i);

sem\_post(&zero\_sem);

}

}

};

## 方法四：原子操作

## 方法五：异步