Lesson 23

1.面向对象和基于对象编程的本质区别在于，基于对象将面向对象中由用户去继承实现的虚函数，改为回调函数，用户只需要向类中注册回调函数即可。

2.资源就是，有些东西从系统中申请，使用完毕后，必须还给系统，否则造成不可预料的后果。（内存、文件描述符、网络连接、数据库连接）。

3.凡是涉及到系统资源的类，一概禁用值语义。

4.线程池内线程的回调函数 runInThread 是 ThreadPool 的成员函数，这里的好处是：

a) 因为是类的内部，所以可以直接访问 isStarted，方便终止线程

```
while(isStarted_)
{
    取任务
    执行任务
}
```

b)可以直接调用 getTask。不必再去指定是哪一个线程池

5.智能指针和 MutexLockGuard 属于 RAII 技术，RAII 即“资源获取即初始化”，资源的获取放在构造函数里面，资源的释放写在析构函数中，这样资源的获取与释放与对象的生存

期是相一致的。

6.Mutex 的使用原则：

a) 使用 RAII 封装 MutexLock 的创建、销毁、加锁和解锁。
b) 只是用互斥锁
c) 绝对不要手工调用 lock 和 unlock，而是改用 MutexLockGuard 来代替，自动化完成加锁、解锁过程
d) 每次加锁时，考虑调用栈上是否已经加锁（例如生产者消费者问题中，Buffer 类中判断队列是否为空）

7.Condition 的使用原则，对于 wait 端：

a) 必须与 Mutex 配合使用，调用 wait 前必须加锁
b) 把对布尔表达式的判断，放在 while 中，而不是 if

```
while(Q.empty())
    full.wait();
```

8.对于 notify 端：

a) notify 不一定是在上锁的情况下调用
b) 进行 notify 前一定要修改了某些条件，例如生产者把产品放入队列，或者消费者取走产品，总之，对布尔表达式的判断造成了一定的影响。
c) 注意区分 notify 和 notifyAll，notify 通常表示资源可用，而 notifyAll 通常用于状态的改变，例如线程池的关闭。

9.子序列的概念：

a) 子序列中的元素都是原字符串中的元素

b) 子序列中元素的排列顺序，与他们在原字符串中的顺序相一致。

c) 例如“foobar”，“fba”就是一个子序列

d) 子序列不同于子串，元素在原字符串之间未必是连续的。

10.动态规划特征

a) 最优子结构，求问题的解必须获取子问题的最优解。

b) 重叠子问题，使用原始的递归存在大量的重复计算。

11.多线程程序尽可能避免使用信号。

12.timerfd 系列的定时器，采用的不是信号，而是 fd 可读，这样就可以把该 fd 加入到 IO 复用模型中。

13.poll 采用的是 LT 水平触发模式，对于某 fd 可读，如果不做 read 处理，那么下次 poll 会再次返回该 fd。

14.timerfd 与 poll 结合时，如果不读取 timerfd 的内容，那么 poll 被一直触发，达不到定时的效果。

15.Timer 与 Thread 的编写：

a) 把用户逻辑 bind 到 Timer 里面

b) 把 Timer 的 runTimer 方法绑定到 Thread 里面