**什么是线程池？**

线程池ThreadPool 就是为了存放“线程”的对象池，它的引入就是为了：限制应用程序中同一时刻运行的线程数。根据系统的环境，可以手动或者自动设置线程数量，达到最优效果。

在程序中，如果某个创建某种对象所需要的代价太高，同时这个对象又可以反复使用，那么我们往往就会准备一个容器，用来保存一批这样的对象。于是乎，我们想要用这种对象时，就不需要每次去创建一个，而直接从容器中取出一个现成的对象就可以了。由于节省了创建对象的开销，程序性能自然就上升了。这个容器就是“池”。很容易理解的是，因为有了对象池，因此在用完对象之后必须有一个“归还”的动作，这样便可以把对象放回池中，下次需要的时候就可以再次拿出来使用了。

对于线程是稀缺资源，创建和销毁开销比较大。可以在一开始就直接创建一批线程对象放在线程池中，然后需要用到线程对象的时候从线程池中取，线程对象用完了放入线程池中。在线程池内部，任务被插入到一个task队列（任务队列，这个在线程池的实现过程中都会用到的数据结构，可以看看网络上每一个线程池的例子都会包括这个任务队列）上，线程池的线程会去取这个队列的任务。当一个新任务插入到队列时，一个空闲线程就会从任务队列上取job。这是最直观的线程池工作过程。

线程池进程用在多线程服务器上，每个通过网络到达服务器的链接都被包装成一个任务并且传递给线程池。线程池的线程会并发的处理连接上的请求。

**线程池的作用：**

合理利用线程池能够带来三个好处。  
第一：降低资源消耗。通过重复利用已创建的线程降低线程创建和销毁造成的消耗。  
第二：提高响应速度。当任务到达时，任务可以不需要等到线程创建就能立即执行。  
第三：提高线程的可管理性。线程是稀缺资源，如果无限制的创建，不仅会消耗系统资源，还会降低系统的稳定性，使用线程池可以进行统一的分配，调优和监控。但是要做到合理的利用线程池，必须对其原理了如指掌。

线程池实现了系统中线程的数量。根据系统的环境情况，可以自动或手动设置线程数量，达到运行的最佳效果；少了浪费了系统资源，多了造成系统拥挤效率不高。用线程池控制线程数量，其他线程排队等候。一个任务执行完毕，再从队列的中取最前面的任务开始执行。若队列中没有等待进程，线程池的这一资源处于等待。当一个新任务需要运行时，如果线程池中有等待的工作线程，就可以开始运行了；否则进入等待队列。

**线程池的实现：**

需要解决的是这些线程对象是由谁来创建的，这些线程对象是在什么创建的？

Java中jdk1.5之后加入了java.util.concurrent包，封装好了线程池。里面有很多参数和属性，可以编写并发程序。但是c++中，需要封装同步原语，使用系统调用，造个轮子。

使用的同步原语有

pthread\_mutex\_t mutex\_l;//互斥锁

pthread\_cond\_t condtion\_l;//条件变量

使用的系统调用有

pthread\_mutex\_init();

pthread\_cond\_init();

pthread\_create(&thread\_[i],NULL,threadFunc,this)

pthread\_mutex\_lock()

pthread\_mutex\_unlock()

pthread\_cond\_signal()

pthread\_cond\_wait()

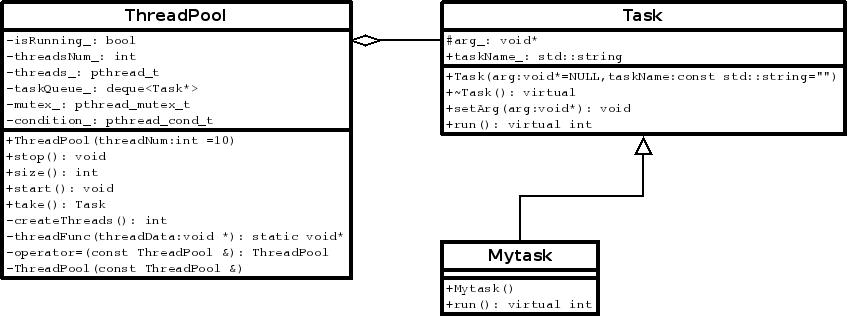
pthread\_cond\_broadcast();

pthread\_join()

pthread\_mutex\_destory()

pthread\_cond\_destory()

实现的思路--->就是怎么组装这些函数和原语：



在ThreadPool中封装基础原语和任务队列，利用面向对象的方法，具体的Mytask任务派生自Task，可以利用多态的机制向ThreadPool中添加一些任务。