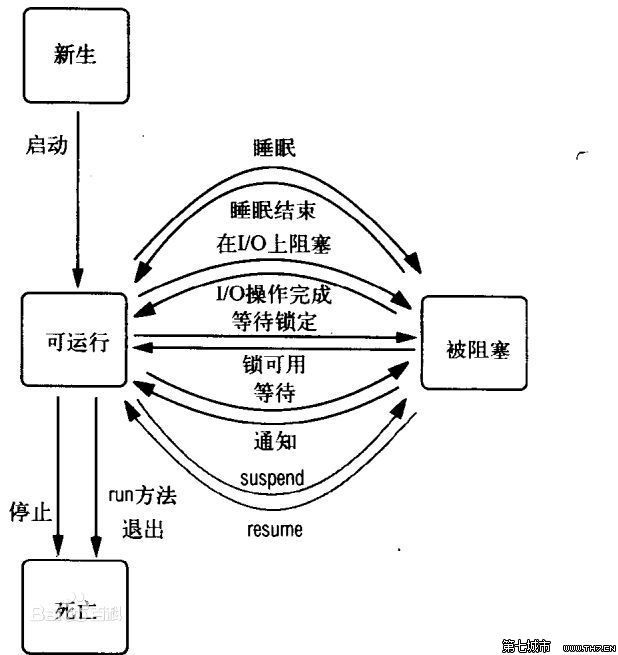
一、基础知识

1、线程的基本概念、线程的基本状态及状态之间的关系

线程是程序执行流的最小单元。线程的基本状态包括：新线程态、可运行态、阻塞/非运行态、死亡态。

各状态之间的关系如下：



2、线程与进程的区别？

线程是指进程内的一个执行单元,也是进程内的可调度实体。与进程的区别:

（1）地址空间:进程内的一个执行单元;进程至少有一个线程;它们共享进程的地址空间;而进程有自己独立的地址空间；

（2）资源拥有:进程是资源分配和拥有的单位,同一个进程内的线程共享进程的资源；

（3）线程是处理器调度的基本单位,但进程不是；

（4）二者均可并发执行；

3、多线程同步和互斥有何异同，在什么情况下使用、多线程同步与互斥的实现方法

同步是一种特殊的互斥。当访问资源量存在先后的顺序的时候使用同步，当需要独占式访问资源时使用互斥。如一个生产者和多个消费者之间。生产者和消费者之间是同步关系；消费者之间是互斥关系。

多线程同步有如下几种实现方法：事件、信号量

多线程互斥有如下几种实现方法：临界区、事件、信号量、互斥量

4、死锁的概念

死锁:是指两个或两个以上的线程在执行过程中，因争夺资源而造成的一种互相等待的现象，若无外力作用，它们都将无法推进下去。

1、CreateThread与\_beginthreadex本质区别

标准C运行库发明的时候，还没有任何操作系统支持多线程，因此埋下了非线程安全的隐患。

为了解决这个问题，Windows操作系统提供了这样的一种解决方案——每个线程都将拥有自己专用的一块内存区域来供标准C运行库中所有有需要的函数使用。而且这块内存区域的创建就是由C/C++运行库函数\_beginthreadex()来负责的。

因此，如果在代码中有使用标准C运行库中的函数时，尽量使用\_beginthreadex()来代替CreateThread()。

参考：<http://blog.csdn.net/morewindows/article/details/7421759>

2、原子操作

原子操作是指不会被线程调度机制打断的操作；这种操作一旦开始，就一直运行到结束，中间不会有任何 context switch （切换到另一个线程）。

在多线程环境中对一个变量进行读写时，我们需要有一种方法能够保证对一个值的递增操作是原子操作——即不可打断性，一个线程在执行原子操作时，其它线程必须等待它完成之后才能开始执行该原子操作。

Windows提供Interlocked系列函数。Linux中提供了两种形式的原子操作：一种是对整数进行的操作，另一种是对单独的位进行操作。

参考：<http://blog.csdn.net/morewindows/article/details/7429155>

3、临界区

（1）临界区包含初始化化、销毁、进入和离开临界区域四个函数。

（2）临界区可以解决线程的互斥问题，但因为具有“线程所有权”，所以无法解决同步问题。

（3）推荐临界区与旋转锁配合使用。由于将线程切换到等待状态的开销较大，因此为了提高临界区的性能，Microsoft将旋转锁合并到临界区中，这样EnterCriticalSection()会先用一个旋转锁不断循环，尝试一段时间才会将线程切换到等待状态。

参考：<http://blog.csdn.net/morewindows/article/details/7442639>

4、事件

（1）事件是内核对象，事件分为手动置位事件和自动置位事件。事件Event内部它包含一个使用计数（所有内核对象都有），一个布尔值表示是手动置位事件还是自动置位事件，另一个布尔值用来表示事件有无触发。

（2）事件可以由SetEvent()来触发，由ResetEvent()来设成未触发。还可以由PulseEvent()来发出一个事件脉冲。

（3）事件可以解决线程间同步问题，因此也能解决互斥问题。

5、互斥量

（1）互斥量是内核对象，它与临界区都有“线程所有权”所以不能用于线程的同步。

（2）互斥量能够用于多个进程之间线程互斥问题，并且能完美的解决某进程意外终止所造成的“遗弃”问题。

四种进程或线程同步互斥的控制方法  
1、临界区:通过对多线程的串行化来访问公共资源或一段代码，速度快，适合控制数据访问。   
2、互斥量:为协调共同对一个共享资源的单独访问而设计的。   
3、信号量:为控制一个具有有限数量用户资源而设计。   
4、事 件:用来通知线程有一些事件已发生，从而启动后继任务的开始。