多线程支持和线程安全（1） PDF转换可能丢失图片或格式，建议阅读原文
https://www.100test.com/kao_ti2020/141/2021_2022__E5_A4_9A_E7_BA_BF_E7_A8_8B_E6_c29_141240.htm 多线程编程方法是一种高级的编程技术，同时它也是复杂应用程序系统中，一种重要的和必不可少的程序设计技术，它能有效地解决各种并行的复杂工作任务，使一些特别复杂的应用系统的总体设计和解决方案，变得更简洁和更清晰起来。多线程编程方法的运用，一般需要位于计算机底层的操作系统的有效支持，同时也需要操作系统能够提供，一些用于多线程支持的系统功能服务接口（如信号量通信原语的操作接口）。但是仅有这些仍然是不够的，例如，这些服务接口太过于底层而不容易被学习使用，也不安全。因此在基础库系统中，同样也需要对多线程编程方法提供更多和更有效的支持。这主要包含两个方面的内容，首先就是提供对多线程支持得更好和更完善的功能封装，也即线程库；另外一项当然也很重要，那就是基础库系统中各组件模块的线程安全设计。 1、 线程库 线程库是基础库系统中一个重要的组成模块。设计线程库时，应该首先考虑它的模型的建立，以及它应该所涵盖的功能。主要包括有用于线程间互斥和同步控制的功能实现与封装，线程的创建、运行和销毁，以及访问并控制线程的一些运行状态或错误状态等。另外线程库也涉及到，线程之间的数据共享或数据通信等功能的实现。下面我们主要介绍两种有代表性的线程库的实现，也即MFC中的线程库和JDK平台中的线程模型。 MFC中为了实现线程间互斥和同步控制功能，主要提供有四种类型的对象，如图10-57所示，包

括CCriticalSection、CEvent、CMutex以及CSemaphore等。 首先介绍CMutex类型，它是对NT操作系统内核中提供的互斥对象（mutex）的封装。它能够确保线程拥有对单个资源的互斥访问权，也即多线程对共享资源的互斥访问。互斥对象包含一个使用数量，一个线程ID和一个递归计数器。并且线程在等待访问资源时可以设定一个超时值。 CCriticalSection类型是对临界区（critical section）的封装。它的作用与特性和CMutex类似，都是用于线程间对共享资源的互斥访问，但是它与CMutex之间稍有不同的是，它不属于内核对象，而是属于用户方式的对象。所以它的效率要比CMutex高很多，并且因为如此，它也只能被用于线程之间的互斥控制，而不能用于进程之间的互斥。 CEvent是对事件（event）内核对象的封装，它是一个最基本，也最灵活的，被用于线程间实现同步的控制对象。CEvent能够通知一个操作已经完成。它有两种不同类型的事件对象。一种是人工重置的事件，另一种是自动重置的事件。当人工重置的事件得到通知时，等待该事件的所有线程均变为可调度线程。当一个自动重置的事件得到通知时，等待该事件的线程中只有一个线程变为可调度线程。 最后一个就是CSemaphore类型，它是对信号量（Semaphore）内核对象的封装。适用于线程之间的同步控制，它是功能最强、使用最灵活，也最复杂的同步控制对象。除了上面讲述到的几种用于实现线程间互斥和同步的控制对象外，MFC的线程库中还有一个非常重要的类模块，也即CWinThread，它是表示应用程序中的执行线程，管理线程的创建、运行和销毁，以及其它许多方面的功能（如线程优先级、状态等）。 100Test 下载频道开通，各类考试题目直接