第十章 避免活跃性危险

#### 一．死锁

1. 小例子---哲学家进餐问题

2. 概念

当一个线程永远的持有一个锁，而其他线程都尝试获得这个锁时，那么它们将永远被阻塞。

（1）抱死

两个线程互相持有对方需要的锁，而又相互等待对方释放锁。

3. 数据库系统处理死锁

当两个事务出现抱死的时候，数据库服务器会强行终止其中一个事务，让另一个事务完成执行，并且稍后重新执行被强行终止的事务。

4. JVM无法像数据库系统那样解决死锁问题。

锁顺序死锁

1. 两个线程以不同的顺序来获得相同的锁，就可能发生死锁。

2. 如果所有的线程以固定的顺序来获得锁，那么在程序中就不会出现锁顺序死锁问题。

动态的锁顺序死锁

1. 转账问题

在协作对象之间发生的死锁

开放调用

1. 概念

如果在调用某个方法的时候不需要持有锁，那么这种调用就称为开放调用。

资源死锁

#### 二．死锁的避免与诊断

1. 一个程序每次只获得一个锁

2. 如果获得多个锁，则考虑获取锁的顺序

3. 尽可能使用开放调用。

4. 支持定时的锁（显示锁Lock的tryLock()）

5. 通过线程转储信息来分析死锁

#### 三．其他活跃性危险

1. 饥饿

yield()与sleep(0)？

避免使用线程优先级，因为这会增加平台依赖性，并可能导致活跃性问题。在大多数并发程序中，都可以使用默认的线程优先级。

2. 糟糕的响应性

（1） 后台运行程序不会使用户失去响应性，但是cpu密集型的后台任务仍然会有影响。

（2） 不良的锁管理也可能导致糟糕的响应性。例如，某个线程长时间的持有一个锁，并且对每个元素进行计算密集型的处理。

3. 活锁

线程将不断重复执行相同的操作，而且总会失败。

两个过于礼貌的人让路：彼此都让路，然后在另一条路上相遇，不断如此。

两台机器发送消息冲突：一秒后再发，再冲突。