DOI:10.14018/j.cnki.cn13-1085/n.2010.07.066

浅析 Java 多线程中的同步和死锁

Analyzing the Synchronization and Deadlock of Java Multithread

赵元媛 Zhao Yuanyuan
(贵州大学科技学院,贵阳 550003)
(The Science and Technology College of Guizhou University,Guiyang 550003,China)

摘要:多线程在编程社会中是一个相当新的结构,非常强大,可以加大地提高程序的运行效率。Java 是第一个在语言的核心中支持线程,通过多线程的并发运行提高了系统资源的利用率,改善了系统性能,但由于多线程要共享内存资源,为了避免数据资源的使用混乱,必须对线程的同步问题作出解决。本文对 Java 多线程中实现同步的方法和死锁产生原因及避免原则做出了简要分析。

Abstract: Multithread is a relatively new and powerful structure in programming. It can increasingly improve the program efficiency. Java is the first to support thread in the core of language. It improves the utilization of the system resource and performance through concurrency. We must resolve the synchronization of multithread in order to avoid the chaotic date because multithread needs to share memory resources. This article will analyze the method of synchronization and the cause and principle of the deadlock.

关键词:多线程;Java;同步;死锁

Key words: multithread;Java;synchronization;deadlock

中图分类号:TP312 文献标识码:A 文章编号:1006-4311(2010)07-0139-01

1 基于 Java 的多线程

多线程是实现并发机制的一种有效手段,它允许编程语言在程序中并发执行多个指令流,每个指令流都称为一个线程,彼此间相互独立,且与进程一样拥有独立的执行控制,由操作系统负责调度。多个线程的执行是并发的,也就是在逻辑上"同时",而不管是否是物理上的"同时"。如果系统只有一个 CPU,那么真正的同时是不可能的,但是由于 CPU 的速度非常快,用户感觉不到其中的区别,因此给人并发执行的假象。

作为一个完全面向对象的语言,在 Java 中写出包含一个或多个线程的程序是很简单的,只要在想要提供线程的类中使用 java.lang 包中的 Thread 类或 Runnable 接口即可实现线程的功能,所以有两种方法可以创建线程:一种是通过继承 Thread 类来创建线程类;另一种方法是建立一个实现 Runnable 接口的类来运行线程。

2 线程的同步机制

有时多线程使用不当可能会造成数据的混乱,例如,两个线程都要访问同一个共享变量,一个线程读这个变量的值并在这个值的基础上完成某些操作,与此同时,另一个线程改变了这个变量值,但第一个线程并不知道,这就造成了数据混乱。而线程同步就是避免同时访问同一数据造成数据混乱的有效方法。

所谓线程同步就是在执行多线程任务时,一次只能有一个线程访问共享资源,其他线程只能等待,只有当该线程完成自己的执行后,另外的线程才可以进入,支持这种互斥的机制称为监视器(Monitor)。在一段时间内只有一个线程拥有监视器,拥有监视器的线程才能访问相应的资源,并锁定资源不让其他线程访问,所有其他的线程在试图访问被锁定的资源时被挂起,等待监视器解锁。

Java 在同步机制中提供了语言级的支持。所有的 Java 对象都有与它们相关的隐含监视器,而关键字 synchronized 则对象的监视器联系,当某个对象用 synchronized 修饰时,表明该对象在任一时刻只能由一个线程访问。synchronized 关键字包括两种用法:synchronized 方法和 synchronized 块。用 syschronize 来标识的方法或区域即为监视器监视的部分,当有线程进入由 synchronized 修饰的部分时,同类对象的其他线程就不能进入这个部分而必须等待,直到该线程执行完毕。

synchronized 关键字的使用仅仅对于一些较为简单的线程间同步问题比较有效,对于哪些复杂的同步问题,比如带有条件的同步问题,Java 提供了另一种同步机制,使线程可以被阻塞,并在条件允许其继续时被唤醒,这个机制建立在系统根类 Object 的 wait()和 notify()、notifyAll()方法上。wait()和 notify()、notifyAll()方法为在相互独立的多线程间协调事件先后提供了一种通信手段。获得对象监视器的线程可以通过调用该对象的 wait()方法主动释放监视器,等待在该对象的线程等待队列上,此时其他线程可以得到监视器从而访问该对象,之后线程可以通过调用 notify()或 notifyAll()方法来唤醒先前因调用 wait()方法而等待的线程。一般情况下,对于 wait()和 notify()、notifyAll()方法的调用都是根据一定的条件来进行的。而且,wait()和 notify()、notifyAll()方法的特性决定了这一对方法必须在 synchronized 方法或块中调用。

3 线程的死锁问题

多线程在使用互斥机制实现同步时,存在"死锁"的潜在危险。死锁是由于两个或多个线程都无法得到相应的监视器而造成相互等待的现象。例如,在某一多线程的程序中有两个共享资源 A 和 B,并且每一个线程都需要获得这两个资源后才可以执行,这是一个同步问题,但是如果没有合理地安排获取这些资源的顺序,就有可能出现线程 1 已经获取资源 A 的锁,由于某种原因被阻塞,此时线程 2 启动并获得资源 B 的锁,再去获得资源 A 的锁时发现线程 1 已经获取,因此等待线程 1 释放 A 锁。线程 1 从阻塞中恢复以后继续执行,欲获取资源 B 的锁,却发现 B 锁已被线程 2 获得,因此也陷入等待。在这种情况下,程序已无法向前推进,在没有外力的情况下,也不会自动退出,因而造成了严重的死锁问题。导致死锁的根源在于不适当地运用 Synchronized 关键字来管理线程对特定对象的访问,其产生的必要条件有如下几点:

①互斥:就是说多个线程不能同时使用同一资源,比如,当线程 A 使用该资源时,B 线程只能等待 A 释放后才能使用;

②占有等待:就是某线程必须同时拥有 N 个资源才能完成任务,否则它将占用已经拥有的资源直到拥有它所需的所有资源为止,就好像游戏中,必须两个球都拿到了,才能释放;

③非剥夺:就是说所有线程的优先级都相同,不能在别的线程没有释放资源的情况下,夺走其已占有的资源;

④循环等待:就是没有资源满足的线程无限期地等待。

遗憾的是,Java 技术并不在语言级别上支持死锁的避免,因此在编程中必须小心地避免死锁,而避免死锁的有效原则是:

⑤当线程因为某个条件未满足而受阻,不能让其继续占用资源。

⑥如果有多个对象需要互斥访问,应确定线程获得锁的顺序,并保证整个程序以相反的顺序释放锁。

作者简介:赵元媛(1980-),女,山西汾阳人,工学硕士,讲师,主要研究方向为程序设计和网站开发。

参考文献:

[1]张思民.Java 语言程序设计.清华大学出版社,2007 年 2 月.

[2]吴东峥.基于 JAVA 多线程同步机制的研究.新疆石油教育学院学报,2005,8.

[3]王建虹.Java 程序设计.高等教育出版社,2007 年 4 月.

[4] 王金海. 基于 Java 多线程的并发机制的研究和实现. 微机发展,2004,14.