Atitit.软件与编程语言中的锁机制原理attilax总结

[1. 用途 （Db，业务数据加锁，并发操作加锁。 1](#_Toc20834)

[2. 锁得类型 排它锁 “互斥锁 共享锁 乐观锁与悲观锁 1](#_Toc28261)

[2.1. 自旋锁还是信号量 1](#_Toc2774)

[2.2. -自动释放还是手动释放 1](#_Toc19145)

[3. 实现方式，语言方式与库方式 1](#_Toc15369)

[4. Java的锁机制 Synchronized ReentrantLock AtomicInteger 2](#_Toc16943)

[5. C# 锁原理（Monitor类和lock关键词 ReaderWriterLock 2](#_Toc6786)

[6. Ref参考资料 3](#_Toc10215)

# 用途 （Db，业务数据加锁，并发操作加锁。

# 锁得类型 排它锁 “互斥锁 共享锁 乐观锁与悲观锁

## 自旋锁还是信号量

## -自动释放还是手动释放

# 实现方式，语言方式与库方式

**[Java](http://lib.csdn.net/base/17" \o "Java EE知识库" \t "http://blog.csdn.net/vking_wang/article/details/_blank)**.util.concurrent.lock 中的Lock 框架是锁定的一个抽象，它允许把锁定的实现作为 Java 类，而不是作为语言的特性来实现。这就为Lock 的多种实现留下了空间，各种实现可能有不同的调度**[算法](http://lib.csdn.net/base/31" \o "算法与数据结构知识库" \t "http://blog.csdn.net/vking_wang/article/details/_blank)**、性能特性或者锁定语义。

Reen

# Java的锁机制 Synchronized ReentrantLock AtomicInteger

是基于JVM来保证数据同步的，而Lock则是在硬件层面，依赖特殊的CPU指令实现数据同步的

实际JDK中也是通过一个32bit的整数位进行CAS操作来实现的。

需要注意的是，用sychronized修饰的方法或者语句块在代码执行完之后锁自动释放，而是用Lock需要我们**手动释放锁**，所以为了保证锁最终被释放(发生异常情况)，要把互斥区放在try内，释放锁放在finally内！！

 Atomic\* 开头的类基本原理都是一致的, 都是借助了底层硬件级别的 Lock 来实现原子操作的。 Cas原理

# C# 锁原理（Monitor类和lock关键词 ReaderWriterLock

C#提供了2种手工控制的锁

一:  Monitor类

     这个算是实现锁机制的纯正类，在锁定的临界区中只允许让一个线程访问，其他线程排队等待。主要整理为2组方法。

1：Monitor.Enter和Monitor.Exit

微软很照护我们，给了我们语法糖Lock,对的

二：ReaderWriterLock类

    先前也知道，Monitor实现的是在读写两种情况的临界区中只可以让一个线程访问，那么如果业务中存在”读取密集型“操作，就

好比数据库一样，读取的操作永远比写入的操作多。针对这种情况，我们使用Monitor的话很吃亏，不过没关系，ReadWriterLock

就很牛X，因为实现了”写入串行“，”读取并行“。

# Ref参考资料

深入浅出Java并发包—锁机制(一) - 一线天色 天宇星辰的日志 - 网易博客.html

【Java线程】锁机制：synchronized、Lock、Condition - Alpha's 学习笔记 - 博客频道 - CSDN.NET.html

Java常用锁机制简介 - hduhans - 博客园.html

5天不再惧怕多线程——第二天 锁机制 - 一线码农 - 博客园.html

C# 多线程编程之锁的使用【互斥锁(lock)和读写锁(ReadWriteLock)】 - C#编程语言程序开发技术文章\_C#编程 - 红黑联盟.html

锁机制与原子操作 \_第四篇\_ - 逆心 - 博客园.html

简单的JavaScript互斥锁.html

作者:: 绰号:老哇的爪子 （ 全名：：Attilax Akbar Al Rapanui 阿提拉克斯 阿克巴 阿尔 拉帕努伊 ）

汉字名：艾提拉（艾龙），   EMAIL:1466519819@qq.com

转载请注明来源： http://blog.csdn.net/attilax

atiend