文章编号：2096-1472(2018)-02-01-03
DOI:10.19644/j.cnki.issn2096-1472.2018.02.001

基于Java synchronized同步锁实现线程交互

崔 政[1,2]，段利国[1]
(1.太原理工大学，山西 太原 030024；
2.山西省电力公司平顺县供电公司，山西 长治 047400)

摘 要：Java多线程能够提高CPU利用效率，但也容易造成线程不安全、线程死锁等问题。本文详细介绍了Java线程各状态之间的关系及其切换，并用实例展示了使用同步锁synchronized保证同一时刻只有一个线程操作同一资源，使用wait()、notify()切换线程状态保证线程操作的前后顺序实现线程交互。理解Java线程各状态之间的关系及其切换，能帮助用户在使用Java多线程的场景有效避免多线程带来的不安全问题。

关键词：多线程；synchronized；线程交互
中图分类号：TP312 **文献标识码**：A

Realization of Thread Interaction Based on Java Synchronized

CUI Zheng[1,2],DUAN Liguo[1]
(1.*Taiyuan University of Technology*,*Taiyuan* 030024,*China*;
2.*Pingshun County Power Supply Company*,*Shanxi Power Company*,*Changzhi* 047400,*China*)

Abstract: Java multi-threading can improve CPU utilization,but it can also cause problems such as thread insecurity and thread deadlock.This paper introduces in detail the relationship between Java thread states and their transitions.Java synchronized ensures that only one thread operates the same resource at the same time,using wait() and notify() to switch the thread state to ensure that thread interactions are implemented in the context of thread operations.Understanding the relationship between the states of Java threads and their switching can help users avoid effectively the unsafe problems of multi-threading in Java multi-threaded scenarios.

Keywords:multi-threading;synchronized;thread interaction

1 引言(Introduction)

Java多线程的使用提高了CPU的利用效率[1-3]，能够带来更好的用户体验。但是不当的线程使用可能会引起线程不安全，不能显示用户想要的结果，甚至造成线程死锁[4,5]，存在程序无法执行等缺点。充分理解Java线程各状态之间的关系及其切换，能够在各种需要使用Java多线程的场景有效避免多线程带来的不安全问题。本文首先介绍Java创建线程的两种方式，接着介绍线程各状态之间的关系及其切换，最后使用synchronized、wait()和notify()实现线程交互。

2 Java创建线程的方式(The way Java creates threads)

Java主要有两种创建线程的方式。第一种方式是创建继承Thread类，并重写其中run()方法的类。创建该类对象，调用start()方法启动线程，Java虚拟机自动调用run()方法执行业务逻辑。第二种方式是创建实现Runnable接口的类，并重写其中的run()方法。创建Thread类对象调用start()方法启动线程，参数为实现Runnable接口的类对象。对应有两种匿名创建线程并启动的方式，分别为：

new Thread() { public void run() { } }.start();

new Thread(new Runnable() { public void run() { } }).start();

通过第二种方式创建的线程有效避免了Java类只允许单继承的缺点，而且如果多个具有相同代码的线程处理同一资源时能有效实现代码和业务的分离，提高代码可读性。

正确地创建并启动线程是实现线程交互的基础。

3 Java线程各状态之间的关系及其切换(The relationship between Java thread states and their switching)

Java线程主要有创建(new)、就绪(runnabe)、执行(running)、阻塞(blocked)及死亡(dead)五种状态。阻塞又

基金项目：山西省重点资助教学改革项目(面向工程教育专业认证持续改进的课程设计实施方案研究，J2017017).

具体分为三种状态，分别是运行时调用join()、sleep()进入blocked状态，运行时获取锁对象进入lock blocked状态，运行时获取锁对象并调用wait()进入wait blocked状态。

通过new Thread对象创建线程后，该线程进入new状态。调用start()方法启动线程进入runnable状态，获取CPU时间片后才进入运行状态，执行具体的业务逻辑。如果run()方法结束或程序出现未捕获的异常或错误，线程进入dead状态。线程在运行的过程中可能出现四种情况：

(1)CPU时间片用完或者调用yield()方法，线程释放CPU资源，重新进入runnable状态，等待获取CPU时间片。

(2)调用join()和sleep()方法进入blocked状态，sleep()时间结束或被打断，join()中断，I/O完成都会回到runnable状态。

(3)调用wait()进入wait blocked状态，直到另一个线程调用notify()或notifyAll()唤醒该线程，使其进入lock blocked状态，释放同步锁使线程回到runnable状态。

(4)加同步锁synchronized进入lock blocked状态，同步锁被释放后进入runnable状态。

如图1所示，join()、sleep()和yield()方法不需要获取synchronized锁对象就可以执行，wait()、notify()或notifyAll()必须和synchronized一起使用。

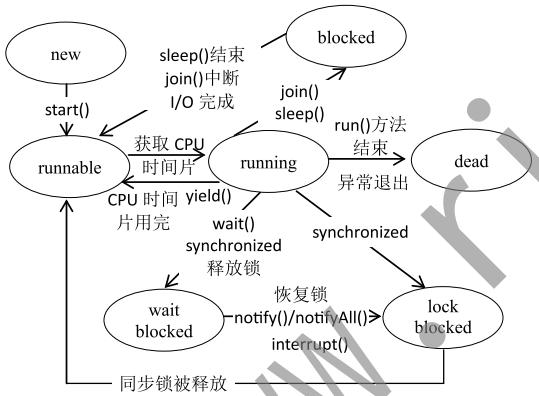

图1 Java线程各状态之间的关系及其切换

Fig.1 The relationship between Java thread states and their switching

4 使用synchronized同步线程(Synchronizing threads with synchronized)

同步锁synchronized可以修饰代码块、方法及静态方法，保证同一时刻最多只有一个线程操作该方法或代码块，避免多个线程同时访问同一公共资源时出现资源违反业务实际场景的线程安全问题[6]。同步锁synchronized实现线程同步的前提是保证多个线程对同一资源的锁对象是相同的。创建资源类Bank，成员变量为金额，成员方法有两个,都是实现每次存100元钱。其中一个为方法锁，另一个为代码块锁。创建两个线程。一个线程调用方法锁进行存钱，另一个线程调用代码块锁进行存钱。当方法锁中的参数为this时，输出用户期望的结果，说明方法锁锁住的是this当前对象。当在该方法锁前面加static修饰时，结果出人意料。修改方法锁的参数为this.getClass()或Bank.class，结果正常。原因是static修饰的方法随着类的加载而加载，属于类的全局方法，被所有对象共享，因此锁住的对象需要是类的全局对象。

5 线程交互(Thread interaction)

定义类Teacher包含属性name和age。类Thread1继承Thread用于给Teacher赋值，根据变量的奇偶赋不同的值。类Thread2继承Thread用于获取Teacher的值，定义测试类。启动两个线程发现结果混乱。使用synchronized将赋值及取值的方法锁住，保证在取的同时不能赋值，在赋值的时候不能获取。因为赋值与取值的操作有先后顺序。在Teacher类中增加变量boolean flag=false判断Teacher中name和age是否为空。两个线程中首先判断标志位flag。如果不满足条件，调用wait()进入等待状态，否则执行具体业务操作。调用notify()唤醒另一个线程，并重新设置flag值，表示可以给Teacher赋值或获取。如下所示为具体程序代码：

```
public class Test4 {public static void main(String[] args) {
    Teacher t=new Teacher();Thread1 t1=new Thread1(t, "线程1");
    Thread2 t2=new Thread2(t,"线程2");t1.start();t2.start();}}
    class Teacher {
        private String name;private int age;
        private boolean flag=false;public Teacher() {}
        public Teacher(String name,int age) {
            super();this.name=name;this.age=age;}
        public String getName() {return name;}
        public void setName(String name) {this.name=name;}
        public int getAge() {return age;}
        public void setAge(int age) {this.age=age;}
        public boolean getFlag() {return flag;}
        public void setFlag(boolean flag) {this.flag=flag;}}
    //赋值Thread1
    class Thread1 extends Thread {Teacher t;public Thread1() {}
        public Thread1(Teacher t,String name) {super(name);this.t=t;}
        int i=0;
        public void run() {while (true) {synchronized (t) {
            if (t.getFlag()) {try {t.wait();} catch (InterruptedException e) {e.printStackTrace();}}
            if (i % 2 == 0) {t.setName("陈易方");t.
```

```
setAge(50);
                } else {t.setName("张芝兰");t.setAge(30);} i++;t.notify();t.setFlag(true); }}}}
    //取值Thread2
    class Thread2 extends Thread {Teacher t; public Thread2() {}
        public Thread2(Teacher t, String name) {super(name);this.t=t;}
        public void run() {while (true) {synchronized (t) {
                if (!t.getFlag()) {try {t.wait();} catch (InterruptedException e) {e.printStackTrace();}}
                System.out.println(t.getName() + "," + t.getAge());t.notify();t.setFlag(false);}}}}
```

Thread1、Thread2中只声明Teacher对象，在构造函数中给Teacher对象赋值，创建Thread对象时可以传入相同的Teacher对象，保证两个线程操作的是同一个Teacher对象。

两个线程同时操作同一个资源时的执行流程如图2所示。在调用start()方法启动两个线程后，这两个线程同时去竞争锁资源。如果线程1竞争到锁资源，线程1判断flag为false，表示能对Teacher赋值，进入执行状态。执行完成，修改flag=true，表示能对Teacher取值，并用notify唤醒等待的线程。如果初始时线程2竞争到锁资源，线程2判断flag=false，表示不能对Teacher取值，进入阻塞状态，直到线程1唤醒。因此初始时只有线程1才能进入运行状态。线程1执行完成，线程2才能进入运行状态，线程1只能等待。线程2执行完成，唤醒正在等待的线程1，线程1再次进入运行状态。如此往复，控制两个线程依次执行，直到程序停止。

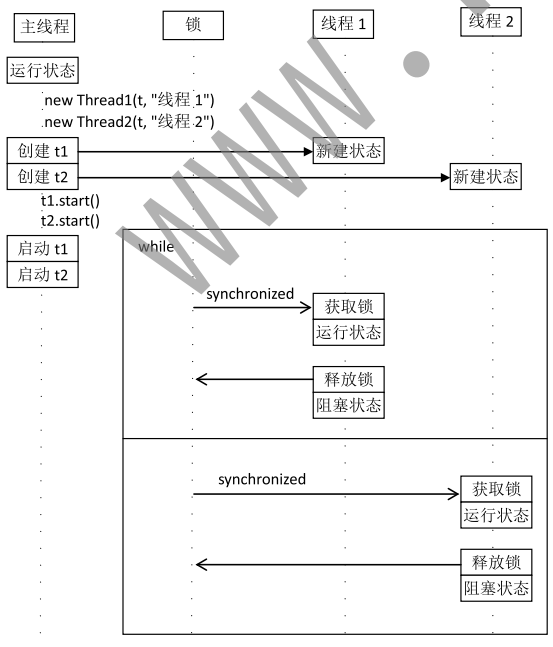

图2 线程交互流程

Fig.2 Thread interaction process

6 结论(Conclusion)

本文首先介绍了创建线程的两种方式，继承Thread类或实现Runnable接口。Java类只允许单继承，Java接口可以多继承多实现。实现Runnable接口创建线程避免了Java类只允许单继承的缺点。接着介绍了Java线程各状态之间的关系及其切换，并详细介绍了synchronized的使用场景及注意事项。最后使用synchronized保证同一时刻只有一个线程操作同一资源，使用wait()、notify()切换线程状态，保证线程操作的前后顺序，实现线程交互。虽然多线程的使用提高了CPU的利用效率，但是线程的切换增加了系统开销，同时也面临着线程不安全、线程死锁的缺点，因此要谨慎使用多线程。对于单线程能处理的问题尽量不要使用多线程。

计算机科学与技术专业的培养计划中大多包括综合性的课程设计，用以培养学生的工程实践能力[7]。侧重软件开发的课程设计过程中，经常出现线程互锁问题。因为学生缺乏实际开发经验，一般很难排除这种故障，从而严重影响学生学习的兴趣和课程设计精度，希望本文内容能对学生有所帮助。

参考文献(References)

[1] A Saxen,A Sanya,G Kuma,et al.A Framework to protect multiple applications in java using synchronization[C].ICTCS '16 Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies,2016(131):70-85.

[2] Tarvo Alexander,Reiss Steven P.Automatic performance prediction of multithreaded programs:a simulation approach[J].Automated Software Engineering,2017(5):1-55.

[3] Cao Man,Zhang Minjia,Sengupta Aritra.Drinking from both glasses:Combining pessimistic and optimistic tracking of cross-thread dependences[J].Acm Sigplan Notices,2016(8):1-13.

[4] 武瑞婵,邓华丽.基于Java多线程的预处理迭代并行求解器[J].山西大同大学学报(自然科学版),2017(2):9-11.

[5] 吴俞伯,郭俊霞,李征,等.基于并发程序数据竞争故障的变异策略[J].计算机应用,2016(11):3170-3177.

[6] 张杨,张冬雯,仇晶.面向Java锁机制的字节码自动重构框架[J].计算机科学,2015(11):84-89.

[7] 段利国,王晓霞,李爱萍,等.面向工程教育持续改进的课程设计实施方案研究[J].软件工程,2017(8):23-27.

作者简介：

崔　政(1992-)，男，硕士生.研究领域：电力系统应用软件开发.

段利国(1970-)，男，博士，副教授.研究领域：智能信息处理.