多线程技术的实现与研究

林恒建
（福建船政交通职业学院信息工程系　福建 福州　350007）

【摘　要】：Java 语言对应用程序多线程的支持，增强了 Java 作为网络程序设计语言的优势，提供了方便用户使用的多任务并发执行的方法。

【关键词】：Java 语言、多线程、同步、状态控制。

1.引言

Java 程序设计语言将线程支持与语言运行环境结合在一起，提供了方便用户使用的多任务并发执行的能力。处在可执行状态中的应用程序称为进程。从用户角度，进程是应用程序的一个动态执行过程。从操作系统角度，进程代表的是操作系统分配的内存、CPU 时间片等资源的基本单位，为正在运行的程序提供的运行环境。进程与应用程序的区别在于应用程序是一段静态代码存储在硬盘等存储空间中，而进程则是处于动态条件下由操作系统维护的系统资源管理实体。线程是比进程更小的执行单元，一个进程可以分为多个线程，形成多条执行路线，完成多项操作。

2.线程的状态控制

在线程的生命周期中，可以将线程分为创建、就绪、运行、睡眠、挂起和死亡六种状态。这些线程状态表明线程处于生命周期的那个阶段。

创建状态：当创建线程对象实例后，系统没有为其分配 CPU 时间片等线程运行资源。就绪状态：在处于创建状态的线程中调用 start()方法，线程的状态转换为就绪状态。此时线程已经获得除 CPU 时间之外的其它系统资源，只等 JVM 的线程调度器按照线程的优先级对该线程进行调度，从而使该线程拥有能够获得 CPU 时间片的机会。睡眠状态：在线程运行过程中调用 sleep()方法并在方法参数中指定线程的睡眠时间，将线程状态转换为睡眠状态。此时线程停止运行指定的睡眠时间，但在不释放占用资源。睡眠时间到达后，线程重新由 JVM 线程调度器进行调度和管理。挂起状态：通过调用 suspend () 方法将线程的状态转换为挂起状态。此时线程释放占用的所有资源，由 JVM 调度转入临时存储空间，当应用程序调用 resume()方法是，恢复线程运行。死亡状态：线程正常运行结束，调 stop()或 destroy()方法后线程进入死亡状态。

3.线程的创建和启动

JAVA 多线程实现方式主要有三种：继承 Thread 类、实现 Runnable 接口,这两种方式线程执行完后都没有返回值；还有一种锁使用 ExecutorService、Callable、Future 实现有返回结果的多线程。

继承 Thread 类的多线程程序设计方法：Thread 类是 JDK 中定义的用于控制线程对象的类，在该类中封装了用于进行线程控制的方法。继承 Thread 类的多线程程序设计方法是让应用程序类继承 Thread 类，并重写 run 方法实现并发性处理过程。启动线程的唯一方法就是通过 Thread 类的 start()实例方法。start()方法是一个 native 方法，它将启动一个新线程，并执行 run()方法。例如：

```
public class ThreadDemo extends Thread {
    public void run() {
     System.out.println("ThreadDemo.run()线程运行中");
    }
}
```

启动线程：

```
ThreadDemo myThread = new ThreadDemo();
myThread.start();
```

实现 Runnable 接口方式实现多线程：

如果自己的类已经继承了另一个类，由于 Java 的单继承机制，无法再继承 Thread 了，此时可通过实现 Runnable 接口方式实现多线程，例如：

```
public class ThreadDemo extends OtherClass implements Runnable {
    public void run(){
     System.out.println("ThreadDemo.run()线程运行中");
    }
}
```

启动 ThreadDemo，需要首先实例化一个 Thread，并传入 ThreadDemo 实例：

```
ThreadDemo threadDemo = new ThreadDemo();
Thread myThread= new Thread(threadDemo);
myThread.start();
```

特别地，可以通过以下两段简洁代码开始一个新线程：

```
new Thread(new Runnable() {
            public void run() {
            //这里写代码,开始一个新线程
            }
        }
        ).start();
```

和

```
new Thread() {
        public void run() {
```

```
            //这里写代码,开始一个新线程
            }
    }.start();
```

第一段代码本质上是通过实现 Runnable 接口方式实现多线程;第二段代码本质上是通过继承 Thread 类实现多线程。可见 Java 程序设计语言对线程的支持,提供了方便用户使用的多任务并发执行的方法。

4.线程的同步

Java 应用程序的多个线程共享同一进程的数据资源,多个线程在并发运行过程中可能同时访问敏感性的内容或访问数量有限资源。在 Java 中定义了线程同步的概念,实现对共享资源的一致性维护。那么应该如何控制呢? 一个比简单有效的办法就是采用信号量机制。信号量机制是定义一个信号量,表示系统能够提供的资源数量;当一个线程占用了一个资源时,信号量减一;当一个线程释放资源时,信号量加一,采用这种方法控制线程访问数量有限资源的同步问题,代码如下:

```
class Semaphore {
private int count;
public Semaphore(int count) {
  //设置资源数量
this.count = count;
}
//用关键字 synchronized 修饰,该方法同一时刻只能由一个线程访问。
public synchronized void acquire() {
while(count == 0) {
try {
//等待资源,由锁申请队列进入锁等待队列。
this.wait();
} catch (InterruptedException e) {
}
}
//获得资源
……
count--;
}
public synchronized void release() {
count++;
  //释放资源
……
  //通知锁等待队列中第一个线程进入锁等待队列。
this.notify();
}
}
```

在 Semaphore 类方法定义中加上用于标识同步方法的关键字 synchronized。在同步方法执行过程中该方法涉及的共享资源如: 上述代码中 count 成员变量将被加上共享锁,确保在方法运行期间只有该方法才能访问共享资源,直到执行该方法的线程运行结束打开共享锁,其它线程才能够访问这些共享资源。

5.总结

如果应用程序必须等待用户输入、等待网络连接、数据库连接、网上数据的传输等数据传输速度较慢的资源和执行时间较长的任务时,多线程应用程序是非常有用的。基于 Internet 的应用程序一般是多线程的,通常的服务器端应用程序必需支持大量客户访问,将服务器端应用程序创建成多线程程序,每个连接用户独占一个线程,缩短服务器响应时间。对执行时间较长的任务也应该放入单独的线程中执行避免用户长时间等待。

参考文献:

[1]埃史尔、陈昊鹏.《Java 编程思想》(第 4 版)北京:机械工业出版社 2007.6

[2] 霍斯特曼 (Cay S.Horstmann)、Gary Cornell、陈昊鹏、王浩.《JAVA 核心技术卷 2:高级特征》机械工业出版社 2008.12

(上接第 90 页)

```
Push to the correspond queue
Pop the data to the initialization queue
```

以学生信息为例,每个学生有对应的信息(专业、班级、姓名、性别、民族、政治面貌、省份)。在乱序的数据中,设置排序的优先级为学生省份、学生班级、学生性别。则该算法处理后的程序会按照设置的优先级分好类。分析本算法的复杂度,只需要 O(kn),k 为需要设置优先级的数据类型,n 为需要分类的数据项。而普通的朴素的算法需要的复杂度为 O(nlogn)。使用类似于基数排序的算法在数据分类上,能比较有效的提高数据分类的效率。

5、结论

完成了一套排序算法的演示程序,可以实际的运用到课堂教学中提高教学效率。同时完成了一个使用基数排序类似算法进行数据分类的程序。

参考文献:

[1]刘红岩,陈剑,陈国青. 数据挖掘中的数据分类算法综述[J]. 清华大学学报(自然科学版),2002,42(6):727-730.

[2]严蔚敏,吴伟民. 数据结构[M]. 北京:清华大学出版社.1997.4

[3]徐章艳,刘作鹏等. 一个复杂度为 max(O(|C||U|,O(|C|2|U/C|)的快速属性约简算法[J]. 计算机学报,2006,29(3):391-399.

[4](美)科曼(Cormen,T.H.) 等著,潘金贵 等译. 算法导论. 机械工业出版社. 2006.9