线程（英语：thread）是操作系统能够进行运算调度的最小单位，线程是独立调度和分派的基本单位。

在Thread类中有：

```
public long getId() {
    return tid;
}

/**
 * A thread state.  A thread can be in one of the following states:
 * <ul>
 * <li>{@link #NEW}<br>
 *     A thread that has not yet started is in this state.
 *     </li>
 * <li>{@link #RUNNABLE}<br>
 *     A thread executing in the Java virtual machine is in this state.
 *     </li>
 * <li>{@link #BLOCKED}<br>
 *     A thread that is blocked waiting for a monitor lock
 *     is in this state.
 *     </li>
 * <li>{@link #WAITING}<br>
 *     A thread that is waiting indefinitely for another thread to
 *     perform a particular action is in this state.
 *     </li>
 * <li>{@link #TIMED_WAITING}<br>
 *     A thread that is waiting for another thread to perform an action
 *     for up to a specified waiting time is in this state.
 *     </li>
 * <li>{@link #TERMINATED}<br>
 *     A thread that has exited is in this state.
 *     </li>
 * </ul>
 *
 * <p>
 * A thread can be in only one state at a given point in time.
 * These states are virtual machine states which do not reflect
 * any operating system thread states.
 *
 * @since   1.5
 * @see #getState
 */
public enum State {
    /**
     * Thread state for a thread which has not yet started.
 */
NEW,

/**
 * Thread state for a runnable thread.  A thread in the runnable
 * state is executing in the Java virtual machine but it may
 * be waiting for other resources from the operating system
 * such as processor.
 */
RUNNABLE,

/**
 * Thread state for a thread blocked waiting for a monitor lock.
 * A thread in the blocked state is waiting for a monitor lock
 * to enter a synchronized block/method or
 * reenter a synchronized block/method after calling
 * {@link Object#wait() Object.wait}.
 */
BLOCKED,

/**
 * Thread state for a waiting thread.
 * A thread is in the waiting state due to calling one of the
 * following methods:
 * <ul>
 *   <li>{@link Object#wait() Object.wait} with no timeout</li>
 *   <li>{@link #join() Thread.join} with no timeout</li>
 *   <li>{@link LockSupport#park() LockSupport.park}</li>
 * </ul>
 *
 * <p>A thread in the waiting state is waiting for another thread to
 * perform a particular action.
 *
 * For example, a thread that has called <tt>Object.wait()</tt>
 * on an object is waiting for another thread to call
 * <tt>Object.notify()</tt> or <tt>Object.notifyAll()</tt> on
 * that object. A thread that has called <tt>Thread.join()</tt>
 * is waiting for a specified thread to terminate.
 */
WAITING,

/**
 * Thread state for a waiting thread with a specified waiting time.
 * A thread is in the timed waiting state due to calling one of
 * the following methods with a specified positive waiting time:
 * <ul>
 *   <li>{@link #sleep Thread.sleep}</li>
```

```
     *   <li>{@link Object#wait(long) Object.wait} with timeout</li>
     *   <li>{@link #join(long) Thread.join} with timeout</li>
     *   <li>{@link LockSupport#parkNanos LockSupport.parkNanos}</li>
     *   <li>{@link LockSupport#parkUntil LockSupport.parkUntil}</li>
     * </ul>
     */
    TIMED_WAITING,

    /**
     * Thread state for a terminated thread.
     * The thread has completed execution.
     */
    TERMINATED;
}
```

Thread的6种状态

在给定的时间点，线程只能处于一种状态。 这些状态是不反映任何操作系统线程状态的虚拟机状态。

NEW（新建程态）、RUNNABLE（可运行态）、 BLOCKED（阻塞状态）、WAITING（等待状态）、TIMED_WAITING（定时等待状态）、TERMINATED（终止状态）

分别介绍

1 NEW

尚未启动的线程的线程状态。

产生一个Thread对象就生成一个新线程。当线程处于"新线程"状态时，仅仅是一个空线程对象，它还没有分配到系统资源。因此只能启动或终止它。任何其他操作都会引发异常。例如，一个线程调用了new方法之后，并在调用start方法之前的处于新线程状态，可以调用start和stop方法。

触发方式：

```
Thread thread = new Thread();
```

2 RUNNABLE

可运行线程的线程状态。 一个处于可运行状态的线程正在Java虚拟机中执行，但它可能正在等待来自操作系统(如处理器)的其他资源。

start（）方法产生运行线程所必须的资源，调度线程执行，并且调用线程的run（）方法。在这时线程处于可运行态。该状态不称为运行态是因为这时的线程并不总是一直占用处理机。特别是对于只有一个处理机的PC而言，任何时刻只能有一个处于可运行态的线程占用处理 机。Java通过调度来实现多线程对处理机的共享。注意，如果线程处于Runnable状态，它也有可能不在运行，这是因为还有优先级和调度问题。

触发方式：

Thread thread = new Thread();
thread.start();
或
thread.run();

start()和run()方法有什么不同?

由源码我们可以知道：

- 调用start()方法时，先放入线程组，进行异步执行，而不一定是直接进行执行
- 调用run()方法，进行同步执行，即直接执行

3 BLOCKED

等待监视器锁而阻塞的线程的线程状态。 处于阻塞状态的线程正在等待监视器锁进入一个同步的块/方法，或者在调用Object.wait之后重新进入一个同步的块/方法。

触发机制：

Object.wait();
thread.interrupt()

4 WAITING

一个处于等待状态的线程正在等待另一个线程执行一个特定的动作。

触发机制：

Object.wait();
或
thread.join()
或(锁支持)
LockSupport.park()
取消等待：
Object.notify()
或
Object.notifyAll()

5 TIMED_WAITING

具有指定等待时间的等待线程的线程状态。由于调用了下列方法中的一个，并且指定了正等待时间，线程处于定时等待状态

触发机制：

Thread.sleep()
或
Object.wait()
或
Thread.join()
或
LockSupport.parkNanos
或
LockSupport.parkUntil

6 TERMINATED

终止线程的线程状态。 线程已经完成执行。

触发机制：

thread.stop();