第八章 线程

目标

- 线程的概念
- 创建和启动线程
- 线程的基本控制
- 同步
- 线程交互
- 线程和Swing

多任务处理

多任务处理有两种类型：

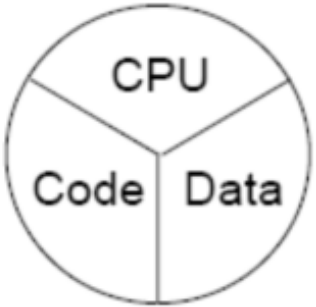
- 基于进程
- 基于线程

进程是指一种“自包容”的运行程序，有自己的地址空间;线程是进程内部单一的一个顺序控制流

基于进程的多任务特点是允许计算机同时运行两个或更多的程序。
基于线程的多任务处理环境中，线程是最小的处理单位。

线程的概念

- 多线程扩展了多任务的概念，一个多线程程序能同时执行多个任务。每个任务称为一个线程（thread），也叫一个执行上下文。
- 一个线程由三个主要部分组成：
  - 一个虚拟CPU
  - 该CPU要执行的代码（Code）
  - 代码操作的数据（Data）



主线程

- 在Java程序启动时，一个线程立刻运行，该线程通常称为程序的主线程。

- 主线程的重要性体现在两个方面：
  - 它是产生其他子线程的线程。
  - 通常它必须最后完成执行，因为它执行各种关闭动作。

基于线程的多任务处理的优点

- 基于线程所需的开销更少
  - 在多任务中，各个进程需要分配它们自己独立的地址空间
  - 多个线程可共享相同的地址空间并且共同分享同一个进程
- 进程间调用涉及的开销比线程间通信多
- 线程间的切换成本比进程间切换成本低

创建线程

- 通过以下两种方法创建 Thread 对象：
  - 声明一个 Thread 类的子类，并覆盖 run() 方法。

```
class MyThread extends Thread {
   public void run( ) {/* 覆盖该方法*/ }
}
```

  - 声明一个实现 Runnable 接口的类，并实现 run() 方法。

```
class MyThread implements Runnable{
  public void run( ) {/* 实现该方法*/ }
}
```

  - 一般使用的比较多的是实现 Runnable 接口，因为Java是单一继承，我们可以实现多个接口

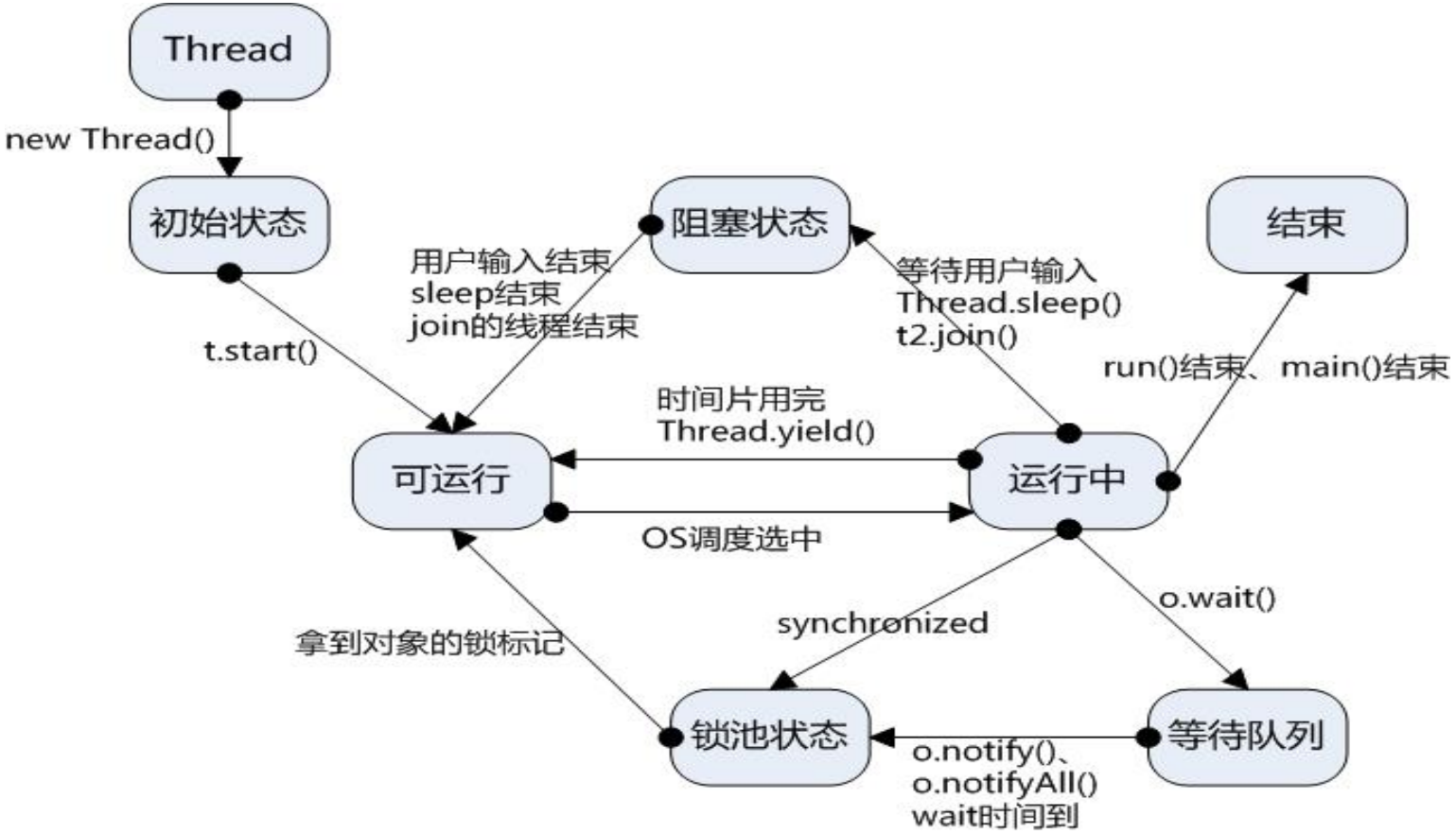
启动线程

- 要触发一个新线程，使用 start() 方法，如：

```
Mythread  t = new Mythread();
t.start();
```

- 在调用 start() 方法时，将创建一个新的控制线程，接着它将调用 run() 方法。
- run() 方法中的代码定义执行线程所需的功能。

线程状态


Thread
new Thread()
初始状态
t.start()
可运行
用户输入结束
sleep结束
join的线程结束
阻塞状态
等待用户输入
Thread.sleep()
t2.join()
结束
run()结束、main()结束
时间片用完
Thread.yield()
运行中
OS调度选中
拿到对象的锁标记
synchronized
o.wait()
锁池状态
o.notify()、
o.notifyAll()
wait时间到
等待队列

线程状态

- **新建状态:**
  - 当使用new操作符创建一个新线程时，如new Thread()，该线程还没有开始运行。它的状态是new
- **可运行状态**
  - 当调用线程的start方法时，线程就处于runnable状态。
- **被阻塞和等待状态**
  - 当一个线程处于被阻塞或等待状态时，它暂时不活动，不运行任何代码。有三种情况决定它达到何种非活动状态：
  - 当一个线程试图获取一个对象锁，而该锁被其他线程持有时，线程进入阻塞状态
  - 当线程等待另一个线程通知调度器一个条件时，它进入等待状态
  - 当调用的方法有超时参数，调用它们将导致线程进入计时等待状态。这一状态将一直保持到超时期限满或接收到合适的通知。

线程状态

- **被终止状态**
  - 程因下面两个原因将被终止：
  - 线程到达其 run() 方法的末尾自然死亡。
  - 线程抛出未捕捉到的异常或错误意外死亡。
  - 您可以调用stop方法杀死一个线程，但是stop方法已过时，不建议使用它。

总结

- 多线程允许程序员编写可最大程度利用CPU 的高效程序。
- Java 以类和接口的形式为多线程提供内置支持。
- Java 程序启动时，一个线程立刻运行，该线程称为主线程。
- 可通过两种方式创建线程：继承Thread类、实现Runnable 接口。

400-133-0510

www.igeekhome.com