# 多线程

## 线程基础

### 线程的应用场景

1）通过并行计算提高程序应用

2）等待网络、IO响应导致的耗时问题

### Java中实现线程的方式

#### 继承Thread

#### 实现Runnable

#### 使用Callable.future

|  |
| --- |
| **public class** CallableDemo **implements** Callable<String> {  **public static void** main(String[] args) **throws** ExecutionException, InterruptedException {  ExecutorService es = Executors.*newCachedThreadPool*();  CallableDemo cd = **new** CallableDemo();  Future<String> future = es.submit(cd);  String str = future.get();*//阻塞* System.***out***.println(str);  es.shutdown();  }  @Override  **public** String call() **throws** Exception {  **return "xxxx"**;  } } |

### 线程的状态

New：没有调用start方法

Runnable ：运行状态

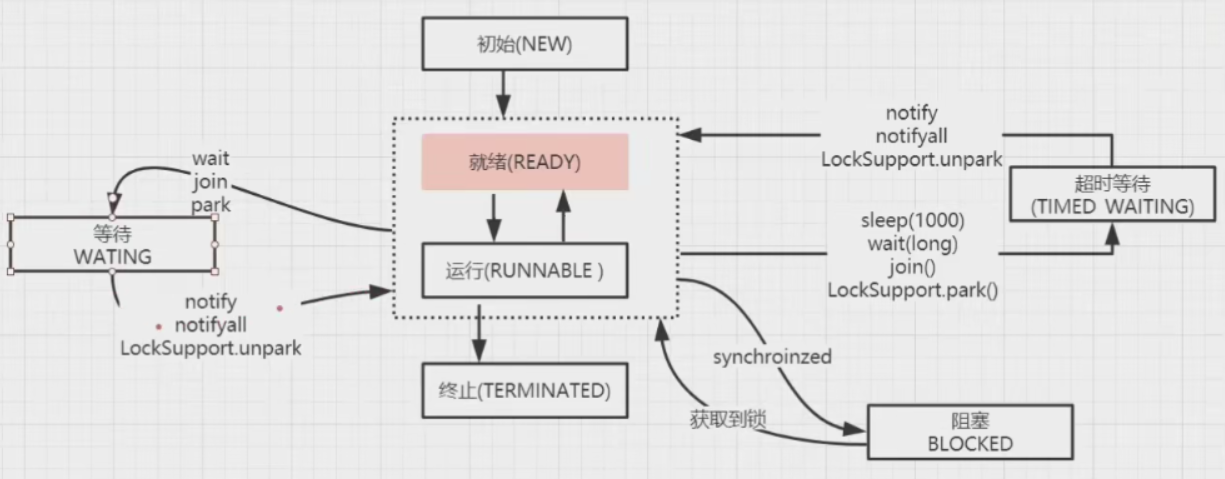
blocked ：阻塞

1. 等待阻塞 wait()
2. 同步阻塞 synchronized
3. 其它阻塞 sleep(); join()

waiting ：等待

timed\_waiting ：时间等待

Terminated：终止



|  |
| --- |
| **public class** ThreadStatusDemo {  **public static void** main(String[] args) {  **new** Thread(  ()->{  **while** (**true**) {  **try** {  TimeUnit.***SECONDS***.sleep(100);  } **catch** (InterruptedException e) {  e.printStackTrace();  }  }  },**"timewaiting"**).start();  **new** Thread(  ()->{  **while** (**true**) {  **synchronized** (ThreadStatusDemo.**class**) {  **try** {  ThreadStatusDemo.**class**.wait();  } **catch** (InterruptedException e) {  e.printStackTrace();  }  }  }},**"waiting"**).start();  **new** Thread(**new** BlockDemo(), **"block-0"**).start();  **new** Thread(**new** BlockDemo(), **"block-1"**).start();  }   **static class** BlockDemo **extends** Thread {  **public void** run() {  **synchronized** (BlockDemo.**class**) {  **while** (**true**) {  **try** {  TimeUnit.***SECONDS***.sleep(100);  } **catch** (InterruptedException e) {  e.printStackTrace();  }  }  }  }  } } |

1. 线程的启动和终止

启动：start

终止：

a.interrupt 优雅的停止线程

|  |
| --- |
| **public class** InterruptDemo {  **static int** *i* = 1;  **public static void** main(String[] args) **throws** InterruptedException {   Thread thread = **new** Thread(  ()->{  **while**(!Thread.*currentThread*().isInterrupted()) {  *i*++;  }  System.***out***.println(*i*);  }, **"interruptdemo"** );  thread.start();  TimeUnit.***SECONDS***.sleep(1);  thread.interrupt();  } } |

b.指令方式，volatile boolean isStop = flase;