[Java中怎么控制线程访问资源的数量](http://blog.csdn.net/u012516914/article/details/40798287)

标签： [Semaphore](http://www.csdn.net/tag/Semaphore)[java资源下载](http://www.csdn.net/tag/java%e8%b5%84%e6%ba%90%e4%b8%8b%e8%bd%bd)[信号量控制](http://www.csdn.net/tag/%e4%bf%a1%e5%8f%b7%e9%87%8f%e6%8e%a7%e5%88%b6)

2014-11-04 23:30 903人阅读 [评论](http://blog.csdn.net/u012516914/article/details/40798287#comments)(0) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/u012516914/article/details/40798287#report)

 分类：



线程（8）



版权声明：本文为博主原创文章，未经博主允许不得转载。

在API中是这样来描述Semaphore 的

Semaphore 通常用于限制可以访问某些资源（物理或逻辑的）的线程数目。

一个计数信号量。从概念上讲，信号量维护了一个许可集。如有必要，在许可可用前会阻塞每一个 [acquire()](http://blog.csdn.net/u012516914/article/details/40798287)，然后再获取该许可。每个 [release()](http://blog.csdn.net/u012516914/article/details/40798287) 添加一个许可，从而可能释放一个正在阻塞的获取者。但是，不使用实际的许可对象，Semaphore 只对可用许可的号码进行计数，并采取相应的行动。

例如，下面的类使用信号量控制线程并发的数量

**[java]** [view plain](http://blog.csdn.net/u012516914/article/details/40798287) [copy](http://blog.csdn.net/u012516914/article/details/40798287)



1. **import** java.util.concurrent.ExecutorService;
2. **import** java.util.concurrent.Executors;
3. **import** java.util.concurrent.Semaphore;

6. **public** **class** TestSemaphore {
8. /\*\*
9. \* @param args
10. \*/
11. **public** **static** **void** main(String[] args) {
12. ExecutorService pool =  Executors.newCachedThreadPool();
13. **final** Semaphore sp = **new** Semaphore(3,**true**);
14. **for**(**int** i=0;i<10;i++){
15. Runnable runnable = **new** Runnable() {
17. @Override
18. **public** **void** run() {
20. **try** {
21. sp.acquire();
22. } **catch** (InterruptedException e) {
23. e.printStackTrace();
24. }
25. System.out.println(sp.availablePermits());
26. System.out.println("线程  "+ Thread.currentThread().getName() +"进入，已有"+ (3-sp.availablePermits())+ "并发") ;
27. **try** {
28. Thread.sleep((**long**) (Math.random()\*3000));
29. } **catch** (InterruptedException e) {
30. e.printStackTrace();
31. }
32. System.out.println("线程  "+Thread.currentThread().getName() +"即将离开 " );
33. sp.release();
34. System.out.println("线程  "+Thread.currentThread().getName() +"离开 ，已有"+ (3-sp.availablePermits()) + "并发");
35. }
36. };
37. pool.execute(runnable);
38. }
39. }
41. }

再例如可以通过信号量来控制线程访问资源：

**[java]** [view plain](http://blog.csdn.net/u012516914/article/details/40798287) [copy](http://blog.csdn.net/u012516914/article/details/40798287)



1. **import** java.util.concurrent.Semaphore;

4. **public** **class** DownloadThread {
5. **private** **static** **int** in\_index = 0;
6. **private** **static** **int** out\_index = 0;
7. **private** **static** **int** buffer\_count = 100;
8. **public** **static** **boolean** g\_downloadComplete;
9. **private** **static** Semaphore g\_seFull = **new** Semaphore(0);
10. **private** **static** Semaphore g\_seEmpty = **new** Semaphore(buffer\_count);
11. **public** **static** **boolean** getBlockFromNet(**int** in\_index) {
12. **int** i = 0;
13. **while** (i < 10000)
14. i++;
15. **if** (in\_index < buffer\_count - 1)
16. **return** **false**;
17. **else**
18. **return** **true**;
19. }
20. **public** **static** **void** writeBlockToDisk(**int** out\_index) {
21. **int** i = 0;
22. **while** (i < 100000)
23. i++;
24. }
26. /\*\*
27. \* @param args
28. \*/
29. **public** **static** **void** main(String[] args) {
30. g\_downloadComplete = **false**;
31. Thread threadA = **new** Thread() {
32. **public** **void** run() {
33. proA();
34. }
35. };
36. Thread threadB = **new** Thread() {
37. **public** **void** run() {
38. proB();
39. }
40. };
41. threadB.start();
42. threadA.start();
43. }


47. **public** **static** **void** proA(){
48. **while** (g\_seFull.availablePermits() < buffer\_count) {
49. **try** {
50. g\_seEmpty.acquire();
51. } **catch** (InterruptedException e1) {
52. // TODO Auto-generated catch block
53. e1.printStackTrace();
54. }
55. g\_downloadComplete = getBlockFromNet(in\_index);
56. in\_index = (in\_index + 1) % buffer\_count;
57. g\_seFull.release();
58. System.out.println("download a block " + in\_index);
59. **if** (g\_downloadComplete)
60. **break**;
61. }
62. }
64. **public** **static** **void** proB(){
65. **while** (g\_seEmpty.availablePermits() > 0) {
66. **try** {
67. g\_seFull.acquire();
68. } **catch** (InterruptedException e1) {
69. // TODO Auto-generated catch block
70. e1.printStackTrace();
71. }
72. writeBlockToDisk(out\_index);
73. out\_index = (out\_index + 1) % buffer\_count;
74. g\_seEmpty.release();
75. System.out.println("write a block " + out\_index);
76. **if** (g\_downloadComplete && out\_index == in\_index)
77. **break**;
78. }
79. }
81. }

