**POCO C++库学习和分析 -- 线程 （三）**

**4. 定时器**

定时器作为线程的扩展，也是编程时经常会被用到的元素。在程序设计上，定时器的作用是很简单。预定某个定时器，即希望在未来的某个时刻，程序能够得到时间到达的触发信号。

编程时，一般对定时器使用有下面一些关注点：

1. 定时器的精度。Poco中的定时器精度并不是很高，具体精度依赖于实现的平台(Windows or Linux)

2. 定时器是否可重复，即定时器是否可触发多次。 Poco中的定时器精度支持多次触发也支持一次触发，由其构造函数Timer决定

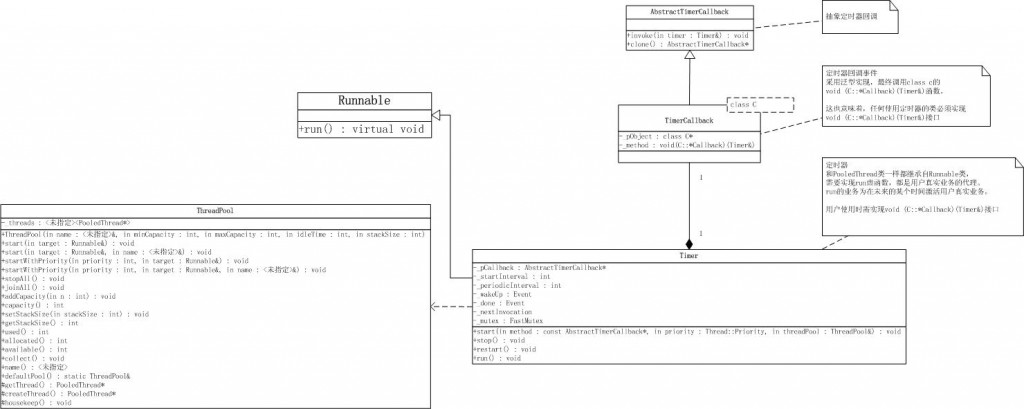
**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8615257)

1. Timer(**long** startInterval = 0, **long** periodicInterval = 0);
2. /// Creates a new timer object. StartInterval and periodicInterval
3. /// are given in milliseconds. If a periodicInterval of zero is
4. /// specified, the callback will only be called once, after the
5. /// startInterval expires.
6. /// To start the timer, call the Start() method.

3. 一个定时器是否可以设置多个时间。 Poco中定时器不支持设置多个时间，每个定时器对应一个时间。如果需要多个时间约定的话，使用者要构造多个定时器。

**4.1 定时器实现**

Poco中的定时器并不复杂，下面是它的类图。



在类图中，Timer继承自Runnable类，也就是说Timer实现了自己的run函数。来看一看，run函数的实现。

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8615257)

1. **void** Timer::run()
2. {
3. Poco::Timestamp now;
4. **long** interval(0);
5. **do**
6. {
7. **long** sleep(0);
8. **do**
9. {
10. now.update();
11. sleep = **static\_cast**<**long**>((\_nextInvocation - now)/1000);
12. **if** (sleep < 0)
13. {
14. **if** (interval == 0)
15. {
16. sleep = 0;
17. **break**;
18. }
19. \_nextInvocation += interval\*1000;
20. ++\_skipped;
21. }
22. }
23. **while** (sleep < 0);
25. **if** (\_wakeUp.tryWait(sleep))
26. {
27. Poco::FastMutex::ScopedLock lock(\_mutex);
28. \_nextInvocation.update();
29. interval = \_periodicInterval;
30. }
31. **else**
32. {
33. **try**
34. {
35. \_pCallback->invoke(\***this**);
36. }
37. **catch** (Poco::Exception& exc)
38. {
39. Poco::ErrorHandler::handle(exc);
40. }
41. **catch** (std::exception& exc)
42. {
43. Poco::ErrorHandler::handle(exc);
44. }
45. **catch** (...)
46. {
47. Poco::ErrorHandler::handle();
48. }
49. interval = \_periodicInterval;
50. }
51. \_nextInvocation += interval\*1000;
52. \_skipped = 0;
53. }
54. **while** (interval > 0);
55. \_done.set();
56. }

在run函数中，我们发现定时器的业务就是不断更新下一次触发时间，并通过睡眠等待到预定时间，触发调用者业务。

**4.2 定时器使用**

最后让我们来看一个定时器的例子：

**[cpp]** [view plaincopy](http://blog.csdn.net/arau_sh/article/details/8615257)

1. #include "Poco/Timer.h"
2. #include "Poco/Thread.h"
3. **using** Poco::Timer;
4. **using** Poco::TimerCallback;
6. **class** TimerExample
7. {
8. **public**:
9. **void** onTimer(Poco::Timer& timer)
10. {
11. std::cout << "onTimer called." << std::endl;
12. }
13. };
15. **int** main(**int** argc, **char**\*\* argv)
16. {
17. TimerExample te;
18. Timer timer(250, 500); // fire after 250ms, repeat every 500ms
19. timer.start(TimerCallback<TimerExample>(te, &TimerExample::onTimer));
20. Thread::sleep(5000);
21. timer.stop();
22. **return** 0;
23. }

（版权所有，转载时请注明作者和出处 <http://blog.csdn.net/arau_sh/article/details/8615257>）