

## 83 基于信号量的线程同步

### 83.1 资源计数

信号量是一种在较多线程瓜分较少资源时使用的停——等机制。一个信号量可以理解为表示某种资源份数的计数器。一个线程在获取一定份数的资源之前，会先尝试对信号量执行相应数值的减法操作。够减说明资源充足，线程可以获得并使用这些资源；不够减说明资源匮乏，线程应放弃对资源的获取，并进入等待状态。得到资源的线程，一旦完成对资源的使用，必须释放所持有的资源，即对信号量执行相应数值的加法操作。这时那些正在等待资源的线程将获得通知，并在对信号量执行减法成功后，拥有并使用相应份数的资源。

### 83.2

在Qt中，通过QSemaphore类表示信号量。该类提供如下方法：

```
1 void QSemaphore::acquire(int n = 1);           // 申请n份资源，阻塞版
2 bool QSemaphore::tryAcquire(int n = 1);       // 申请n份资源，非阻塞版
3 bool QSemaphore::tryAcquire(int n, int timeout); // 申请n份资源，超时版
4 void QSemaphore::release(int n = 1);         // 释放n份资源
5 int QSemaphore::available() const;          // 获取资源的可用份数
```

### 83.4 案例

#### 83.4.1 创建项目

通过QtCreator，在C:\Users\Minwei\Projects\Qt路径下，创建名为Semaphore的控制台（Console）项目。

#### 83.4.2 实现功能

C:\Users\Minwei\Projects\Qt\Semaphore\main.cpp：

```
1 #include <iostream>
2 using namespace std;
3
4 #include <QCoreApplication>
5 #include <QThread>
6 #include <QMutex>
7 #include <QSemaphore>
8 #include <QRandomGenerator>
9
10 #define SEAT_COUNT 3
11 #define PASSENGER_COUNT 9
12
13 QMutex g_mutex;
14 QSemaphore g_seats(SEAT_COUNT);
15
16 class Passenger: public QThread
17 {
18 public:
19     Passenger(QString name)
20         : m_name(name)
21         , m_rand(time(NULL))
```

```

22     {
23     }
24
25 protected:
26     void run()
27     {
28         g_mutex.lock();
29         cout << m_name.toString() << "> STANDING" << endl;
30         g_mutex.unlock();
31
32         g_seats.acquire();
33
34         g_mutex.lock();
35         cout << m_name.toString() << "> SIT DOWN" << endl;
36         g_mutex.unlock();
37
38         msleep(m_rand.bounded(1, 101));
39
40         g_mutex.lock();
41         cout << m_name.toString() << "> GET OFF" << endl;
42         g_mutex.unlock();
43
44         g_seats.release();
45     }
46
47 private:
48     QString m_name;
49     QRandomGenerator m_rand;
50 };
51
52 int main(int argc, char *argv[])
53 {
54     QApplication a(argc, argv);
55
56     cout << g_seats.available() << endl;
57
58     Passenger* passengers[PASSENGER_COUNT];
59
60     for (int i = 0; i < PASSENGER_COUNT; ++i)
61     {
62         passengers[i] = new Passenger(QString("PASSENGER-%1").arg(i + 1));
63         passengers[i]->start();
64     }
65
66     for (int i = 0; i < PASSENGER_COUNT; ++i)
67     {
68         passengers[i]->wait();
69         delete passengers[i];
70     }
71
72     cout << g_seats.available() << endl;
73
74     return a.exec();
75 }
76

```

### 05.4.3 测试验证

运行效果如图所示:

```
1 3
2 PASSENGER-1> STANDING
3 PASSENGER-2> STANDING
4 PASSENGER-3> STANDING
5 PASSENGER-4> STANDING
6 PASSENGER-6> STANDING
7 PASSENGER-1> SIT DOWN
8 PASSENGER-5> STANDING
9 PASSENGER-7> STANDING
10 PASSENGER-8> STANDING
11 PASSENGER-9> STANDING
12 PASSENGER-2> SIT DOWN
13 PASSENGER-3> SIT DOWN
14 PASSENGER-1> GET OFF
15 PASSENGER-4> SIT DOWN
16 PASSENGER-3> GET OFF
17 PASSENGER-2> GET OFF
18 PASSENGER-5> SIT DOWN
19 PASSENGER-6> SIT DOWN
20 PASSENGER-4> GET OFF
21 PASSENGER-8> SIT DOWN
22 PASSENGER-5> GET OFF
23 PASSENGER-6> GET OFF
24 PASSENGER-7> SIT DOWN
25 PASSENGER-9> SIT DOWN
26 PASSENGER-8> GET OFF
27 PASSENGER-7> GET OFF
28 PASSENGER-9> GET OFF
29 3
```